El libro de la implementación de Clientes Ultra Finos

Alberto Castillo
Revisión 1

Guía de implementación:
Entornos gráficos Cliente/Servidor con arquitecturas Intel i486.

Creative Commons <http://creativecommons.org/>

Creative Commons Legal Code

Atribución-NoComercial-CompartirDerivadasIgual 2.5 (Argentina)

CREATIVE COMMONS CORPORATION NO ES UN ESTUDIO JURÍDICO NI PROVEE SERVICIOS LEGALES. LA DISTRIBUCIÓN DE ESTA LICENCIA NO CREA UNA RELACIÓN ABOGADO-CLIENTE. CREATIVE COMMONS PROVEE ESTA INFORMACIÓN "TAL Y COMO SE LA ENCUENTRA". CREATIVE COMMONS NO DA GARANTÍAS EN RELACIÓN A LA INFORMACIÓN PROPORCIONADA Y SE LIBERA DE RESPONSABILIDAD POR LOS DAÑOS QUE RESULTEN DE SU USO.

/Licencia/

LA OBRA (TAL COMO SE DEFINE MÁS ABAJO) SE PROVEE BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS (“CCPL” O “LICENCIA”). LA OBRA ESTÁ PROTEGIDA POR EL DERECHO DE AUTOR Y/O POR OTRAS LEYES APLICABLES. ESTÁ PROHIBIDO CUALQUIER USO DE LA OBRA DIFERENTE AL AUTORIZADO BAJO ESTA LICENCIA O POR EL DERECHO DE AUTOR.

MEDIANTE EL EJERCICIO DE CUALQUIERA DE LOS DERECHOS AQUÍ OTORGADOS SOBRE LA OBRA, USTED ACEPTA Y ACUERDA QUEDAR OBLIGADO POR LOS TÉRMINOS DE ESTA LICENCIA. EL LICENCIANTE LE CONCEDE LOS DERECHOS AQUÍ CONTENIDOS CONSIDERANDO QUE USTED ACEPTA SUS TÉRMINOS Y CONDICIONES.

1. Definiciones

1. *"Obra Colectiva"* significa una obra, tal como una edición periódica, antología o enciclopedia, en la cual la Obra, en su integridad y forma inalterada, se ensambla junto a otras contribuciones que en sí mismas también constituyen obras separadas e independientes, dentro de un conjunto colectivo. Una obra que integra una Obra Colectiva no será considerada una Obra Derivada (tal como se define más abajo) a los fines de esta Licencia.
2. *"Obra Derivada"* significa una obra basada sobre la Obra o sobre la Obra y otras obras preexistentes, tales como una traducción, arreglo musical, dramatización, ficcionalización, versión fílmica, grabación sonora, reproducción artística, resumen, condensación, o cualquier otra forma en la cual la Obra puede ser reformulada, transformada o adaptada. Una obra que constituye una Obra Colectiva no será considerada una Obra Derivada a los fines de esta Licencia. Para evitar dudas, cuando la Obra es una composición musical o grabación sonora, la sincronización de la Obra en una relación temporal con una imagen en movimiento ("synching") será considerada una Obra Derivada a los fines de esta Licencia.
3. *"Licenciante"* significa el individuo o entidad que ofrece la Obra bajo los términos de esta Licencia.
4. *"Autor Original"* significa el individuo o entidad que creó la Obra.
5. *“Obra”* significa la obra sujeta al derecho de autor que se ofrece bajo los términos de esta Licencia.
6. *"Usted"* significa un individuo o entidad ejerciendo los derechos bajo esta Licencia quien previamente no ha violado los términos de

esta Licencia con respecto a la Obra, o quien, a pesar de una previa violación, ha recibido permiso expreso del Licenciante para ejercer los derechos bajo esta Licencia.

7. *"Elementos de la Licencia"* significa los siguientes atributos principales de la licencia elegidos por el Licenciante e indicados en el título de la Licencia: Atribución, NoComercial, CompartirDerivadasIgual.

2. Derechos de Uso Libre y Legítimo. Nada en esta licencia tiene por objeto reducir, limitar, o restringir cualquiera de los derechos provenientes del uso libre, legítimo, derecho de cita u otras limitaciones que tienen los derechos exclusivos del titular bajo las leyes del derecho de autor u otras normas que resulten aplicables.

3. Concesión de la Licencia. Sujeto a los términos y condiciones de esta Licencia, el Licenciante por este medio le concede a Usted una licencia de alcance mundial, libre de regalías, no-exclusiva, perpetua (por la duración del derecho de autor aplicable) para ejercer los derechos sobre la Obra como se establece abajo:

1. para reproducir la Obra, para incorporar la Obra dentro de una o más Obras Colectivas, y para reproducir la Obra cuando es incorporada dentro de una Obra Colectiva;
2. para crear y reproducir Obras Derivadas;
3. para distribuir copias o fonogramas, exhibir públicamente, ejecutar públicamente y ejecutar públicamente por medio de una transmisión de audio digital las Obras, incluyendo las incorporadas en Obras Colectivas;
4. para distribuir copias o fonogramas, exhibir públicamente, ejecutar públicamente y ejecutar públicamente por medio de una transmisión de audio digital las Obras Derivadas;

Los derechos precedentes pueden ejercerse en todos los medios y formatos ahora conocidos o a inventarse. Los derechos precedentes incluyen el derecho de hacer las modificaciones técnicamente necesarias para ejercer los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el Licenciante son reservados, incluyendo, aunque no sólo limitado a estos, los derechos presentados en las Secciones 4 (e) y 4 (f).

*4. **Restricciones*. La licencia concedida arriba en la Sección 3 está expresamente sujeta a, y limitada por, las siguientes restricciones:

1. Usted puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente la Obra en forma digital sólo bajo los términos de esta Licencia, y Usted debe incluir una copia de esta Licencia o de su Identificador Uniforme de Recursos (Uniform Resource Identifier) con cada copia o fonograma de la Obra que Usted distribuya, exhiba públicamente, ejecute públicamente, o ejecute públicamente en forma digital. Usted no podrá ofrecer o imponer condición alguna sobre la Obra que altere o restrinja los términos de esta Licencia o el ejercicio de los derechos aquí concedidos a los destinatarios. Usted no puede sublicenciar la Obra. Usted debe mantener intactas todas las notas que se refieren a esta Licencia y a la limitación de garantías. Usted no puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente en forma digital la Obra con medida tecnológica alguna que controle el acceso o uso de la Obra de una forma inconsistente con los términos de este Acuerdo de Licencia. Lo antedicho se aplica a la Obra cuando es incorporada en una Obra Colectiva, pero esto no requiere que la Obra

Colectiva, con excepción de la Obra en sí misma, quede sujeta a los términos de esta Licencia. Si Usted crea una Obra Colectiva, bajo requerimiento de cualquier Licenciante Usted debe, en la medida de lo posible, quitar de la Obra Colectiva cualquier crédito requerido en la cláusula 4(d), conforme lo solicitado. Si Usted crea una Obra Derivada, bajo requerimiento de cualquier Licenciante Usted debe, en la medida de lo posible, quitar de la Obra Derivada cualquier crédito requerido en la cláusula 4(d), conforme lo solicitado.

2. Usted puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente en forma digital una Obra Derivada sólo bajo los términos de esta Licencia, una versión posterior de esta Licencia con los mismos Elementos de la Licencia, o una licencia de Creative Commons iCommons que contenga los mismos Elementos de la Licencia (v.g., Atribución, NoComercial, CompartirDerivadasIgual 2.5 de Japón). Usted debe incluir una copia de esta licencia, o de otra licencia de las especificadas en la oración precedente, o de su Identificador Uniforme de Recursos (Uniform Resource Identifier) con cada copia o fonograma de la Obra Derivada que Usted distribuya, exhiba públicamente, ejecute públicamente o ejecute públicamente en forma digital. Usted no podrá ofrecer o imponer condición alguna sobre la Obra Derivada que altere o restrinja los términos de esta Licencia o el ejercicio de los derechos aquí concedidos a los destinatarios, y Usted debe mantener intactas todas las notas que refieren a esta Licencia y a la limitación de garantías. Usted no puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente en forma digital la Obra Derivada con medida tecnológica alguna que controle el acceso o uso de la Obra de una forma inconsistente con los términos de este Acuerdo de Licencia. Lo antedicho se aplica a la Obra Derivada cuando es incorporada en una Obra Colectiva, pero esto no requiere que la Obra Colectiva, con excepción de la Obra Derivada en sí misma, quede sujeta a los términos de esta Licencia.
3. Usted no puede ejercer ninguno de los derechos a Usted concedidos precedentemente en la Sección 3 de alguna forma que esté primariamente orientada, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas. El intercambio de la Obra por otros materiales protegidos por el derecho de autor mediante el intercambio de archivos digitales (file-sharing) u otras formas, no será considerado con la intención de, o dirigido a, la obtención de ventajas comerciales o compensaciones monetarias privadas, siempre y cuando no haya pago de ninguna compensación monetaria en relación con el intercambio de obras protegidas por el derecho de autor.
4. Si usted distribuye, exhibe públicamente, ejecuta públicamente o ejecuta públicamente en forma digital la Obra o cualquier Obra Derivada u Obra Colectiva, Usted debe mantener intacta toda la información de derecho de autor de la Obra y proporcionar, de forma razonable según el medio o manera que Usted esté utilizando: (i) el nombre del Autor Original si está provisto (o seudónimo, si fuere aplicable), y/o (ii) el nombre de la parte o las partes que el Autor Original y/o el Licenciante hubieren designado para la atribución (v.g., un instituto patrocinador, editorial, publicación) en la información de los derechos de autor del Licenciante, términos de servicios o de otras formas razonables; el título de la Obra si está provisto; en la medida de lo razonablemente factible y, si está provisto, el Identificador Uniforme de Recursos (Uniform Resource Identifier) que el Licenciante especifica para ser asociado con la Obra, salvo que tal URI no se refiera a la nota sobre los derechos de autor o a la

información sobre el licenciamiento de la Obra; y en el caso de una Obra Derivada, atribuir el crédito identificando el uso de la Obra en la Obra Derivada (v.g., "Traducción Francesa de la Obra del Autor Original," o "Guión Cinematográfico basado en la Obra original del Autor Original"). Tal crédito puede ser implementado de cualquier forma razonable; en el caso, sin embargo, de Obras Derivadas u Obras Colectivas, tal crédito aparecerá, como mínimo, donde aparece el crédito de cualquier otro autor comparable y de una manera, al menos, tan destacada como el crédito de otro autor comparable.

5.

Para evitar dudas, cuando una Obra es una composición musical:

1. * Derechos Económicos y Ejecución bajo estas Licencias.* El Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente o vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, ARGENTORES), los valores (royalties) por la ejecución pública o por la ejecución pública en forma digital (v.g., webcast) de la Obra si esta ejecución está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.
2. * Derechos Económicos sobre Fonogramas.* El Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente, vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, AADI-CAPIF), o vía una agencia de derechos musicales o algún agente designado, los valores (royalties) por cualquier fonograma que Usted cree de la Obra (“versión", "cover”) y a distribuirlos, conforme a las disposiciones aplicables del derecho de autor, si su distribución de la versión (cover) está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.

6. * Derechos Económicos y Ejecución Digital (Webcasting)*. Para evitar dudas, cuando la Obra es una grabación sonora, el Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente o vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, ARGENTORES), los valores (royalties) por la ejecución pública digital de la Obra (v.g., webcast), conforme a las disposiciones aplicables de derecho de autor, si esta ejecución está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.

5. Representaciones, Garantías y Limitación de Responsabilidad

A MENOS QUE SEA ACORDADO DE OTRA FORMA Y POR ESCRITO ENTRE LAS PARTES, EL LICENCIANTE OFRECE LA OBRA “TAL Y COMO SE LA ENCUENTRA” Y NO OTORGA EN RELACIÓN A LA OBRA NINGÚN TIPO DE REPRESENTACIONES O GARANTÍAS, SEAN EXPRESAS, IMPLÍCITAS O LEGALES; SE EXCLUYEN ENTRE OTRAS, SIN LIMITACIÓN, LAS GARANTÍAS SOBRE LAS CONDICIONES, CUALIDADES, TITULARIDAD O EXACTITUD DE LA OBRA, ASÍ COMO TAMBIÉN, LAS GARANTÍAS SOBRE LA AUSENCIA DE ERRORES U OTROS DEFECTOS, SEAN ESTOS MANIFIESTOS O LATENTES, PUEDAN O NO DESCUBRIRSE. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE GARANTÍAS IMPLÍCITAS, POR TANTO ESTAS EXCLUSIONES PUEDEN NO APLICÁRSELE A USTED.

*6. **Limitación de Responsabilidad.* EXCEPTO EN LA EXTENSIÓN REQUERIDA POR LA LEY APLICABLE, EL LICENCIANTE EN NINGÚN CASO SERÁ REPONSABLE FRENTE A USTED, CUALQUIERA SEA LA TEORÍA LEGAL, POR CUALQUIER DAÑO

ESPECIAL, INCIDENTAL, CONSECUENTE, PUNITIVO O EJEMPLAR, PROVENIENTE DE ESTA LICENCIA O DEL USO DE LA OBRA, AUN CUANDO EL LICENCIANTE HAYA SIDO INFORMADO SOBRE LA POSIBILIDAD DE TALES DAÑOS.

7. Finalización

1. Esta Licencia y los derechos aquí concedidos finalizarán automáticamente en caso que Usted viole los términos de la misma. Los individuos o entidades que hayan recibido de Usted Obras Derivadas u Obras Colectivas conforme a esta Licencia, sin embargo, no verán finalizadas sus licencias siempre y cuando permanezcan en un cumplimiento íntegro de esas licencias. Las secciones 1, 2, 5, 6, 7, y 8 subsistirán a cualquier finalización de esta Licencia.
2. Sujeta a los términos y condiciones precedentes, la Licencia concedida aquí es perpetua (por la duración del derecho de autor aplicable a la Obra). A pesar de lo antedicho, el Licenciante se reserva el derecho de difundir la Obra bajo diferentes términos de Licencia o de detener la distribución de la Obra en cualquier momento; sin embargo, ninguna de tales elecciones servirá para revocar esta Licencia (o cualquier otra licencia que haya sido, o sea requerida, para ser concedida bajo los términos de esta Licencia), y esta Licencia continuará con plenos efectos y validez a menos que termine como se indicó precedentemente.

8. Misceláneo

1. Cada vez que Usted distribuye o ejecuta públicamente en forma digital la Obra o una Obra Colectiva, el Licenciante ofrece a los destinatarios una licencia para la Obra en los mismos términos y condiciones que la licencia concedida a Usted bajo esta Licencia.
2. Cada vez que Usted distribuye o ejecuta públicamente en forma digital una Obra Derivada, el Licenciante ofrece a los destinatarios una licencia para la Obra original en los mismos términos y condiciones que la licencia concedida a Usted bajo esta Licencia.
3. Si alguna disposición de esta Licencia es inválida o no exigible bajo la ley aplicable, esto no afectará la validez o exigibilidad de los restantes términos de esta Licencia, y sin necesidad de más acción de las partes de este acuerdo, tal disposición será reformada en la mínima extensión necesaria para volverla válida y exigible.
4. Ningún término o disposición de esta Licencia se considerará renunciado y ninguna violación se considerará consentida a no ser que tal renuncia o consentimiento sea por escrito y firmada por las partes que serán afectadas por tal renuncia o consentimiento.
5. Esta Licencia constituye el acuerdo integral entre las partes con respecto a la Obra licenciada aquí. No hay otros entendimientos, acuerdos o representaciones con respecto a la Obra que no estén especificados aquí. El Licenciante no será obligado por ninguna disposición adicional que pueda aparecer en cualquier comunicación proveniente de Usted. Esta Licencia no puede ser modificada sin el mutuo acuerdo por escrito entre el Licenciante y Usted.

Creative Commons no es una parte para esta Licencia y no da garantía alguna en relación con la Obra. Creative Commons no será responsable bajo ninguna teoría legal, frente a Usted o cualquier parte, de ningún daño en absoluto, excluyendo también, sin limitación, cualquier daño general, especial, incidental o consecuente, originado en relación con esta licencia. No obstante las dos (2) oraciones precedentes, si en virtud de este documento Creative Commons se ha identificado

expresamente como el Licenciante, tendrá todos los derechos y obligaciones del Licenciante.

Excepto con el propósito limitado de indicar al público que la Obra está licenciada bajo la CCPL, ninguna parte usará la marca “Creative Commons”, o ninguna marca o logotipo relacionado a Creative Commons, sin el previo consentimiento por escrito de Creative Commons. Cualquier uso permitido será de conformidad con la guía de uso de la marca Creative Commons entonces vigente, según sea publicada periódicamente en su sitio web o, de otro modo, esté disponible mediante solicitud.

Creative Commons puede ser contactada en http://creativecommons.org/.

Volver al Commons Deed
<http://creativecommons.org/licenses/by-nc-sa/2.0/ar/>

Todos los nombres propios de programas, sistemas operativos, equipos hardware, etc. que aparecen en este libro son marcas registradas de sus respectivas compañías u organizaciones.

Documento creado por Alberto Castillo – 2009/2010
Implementación creada por Alberto Castillo – 2005/2010
Propiedad intelectual de Alberto Castillo
albertocastillo@hotmail.com

Dedicado a:

Mi familia como siempre, a mi mujer y mis hijos, a Edu y Billy que fueron los que me despertaron el interes por la electronica aplicada a las computadoras alla cuando tenia 11 años, a Sinclair e hijos (zx-81 y zx-spectrum) por mostrarme el maravilloso mundo del codigo maquina, a mis antiguos profesores de electronica e informatica de mi querida escuela E.N.E.T. Nro. 2 ING. CARLOS ALBERTO CASSAFFOUSTH y a todas las personas sin títulos académicos como yo, que se preguntan ¿como funcionan las cosas?.

Agradecimientos:

Antes de comenzar quiero agradecer profundamente a Csaba Henk por sus conocimientos y su aporte a la comunidad con sus documentos acerca de su trabajo, el cual es lo suficientemente inspirador para mi, como para considerarlo un héroe, a sus consejos y ayuda personal hacia este proyecto, a Pablo Perez de Angelis por la primer versión de la implementación, la cual, aporta la importante modificación del uso del teclado en español y el arranque automático del cliente VNC, al grupo de desarrolladores y a la comunidad de LTSP, la cual me ha ayudado de forma desinteresada durante estos 5 años, al proyecto Fedora en el cual vengo implementando desde su versión Fedora Core2, a la comunidad Fedora Argentina por todo su apoyo incondicional, a Red Hat por su capacitación RH133, la cual me sirvió para continuar con mi trabajo (recibí una beca gracias a Fedora por el interés que genero mi proyecto en el año 2007), a los responsables de Gleducar que me ayudaron con la difusión de este proyecto, a la comunidad Ubuntu hispana por haber recibido mi documento de implementación de LTSP *"Mini TUTO (alpha todavía) Edubuntu 6.06 + LTSP 4.2.1"* con tanto interés en el año 2006 (el cual a la fecha a sido objeto de mas de 11.000 lecturas), a los desarrolladores LFS y su comunidad LFS, Cross-LFS y LFS-Embedded (Linux From Scratch) por el soporte técnico y la enseñanza aportada en eéstos, mis primeros pasos, especialmente gracias a *Jimg* por la importante documentación sobre sistemas embebidos aportada, a la comunidad K12linux, por el aporte en el desarrollo de distribuciones Fedora y CentOs para la educación inicial y primaria, a las distribuciones que me han acompañado durante todo este tiempo: Fedora, CentOs, Ubuntu y Mandrake, a Fernando Flores de la Dirección de Sistemas del Ministerio de Educación de la Provincia de Córdoba – Argentina, por la invaluable documentación que ha realizado para este proyecto tecnológico/educativo con sus guías "Guía *rápida del usuario OpenOffice*" y "Guía *para planificar la ayuda técnica del laboratorio de informática*" a Marcela Falco también de esta dependencia por la guía de uso básico del entorno gráfico y por ultimo, pero no por eso menos importante, al Ministerio de Educación de la Provincia de Córdoba.

Acerca del autor

Soy Técnico Electrónico, egresado de la escuela E.N.E.T- Nro. 2, Ing. C. A. Cassaffousth en el año 1993.
He trabajado en distintos frentes, desde calibrador de filterplexers y antenista, instalador de sistemas de alarma, incendio, telefonía y cctv, vendedor de mostrador, lavacopas, mozo (camarero), soporte informático, hasta administrador, desarrollador e implementador Junior de sistemas Gnu/Linux.
Actualmente presto servicio en el Ministerio de Educación de la Provincia de Córdoba como soporte técnico y en la SubSecretaría de Planeamiento Físico de la Universidad Nacional de Córdoba como soporte técnico y administrador de su red.
Tambien soy asesor informatico freelance.
Soy músico profesional desde 1992 y mantengo vivo mi pequeño taller de electrónica en casa.

Índice de contenido

Introducción....18
Descripción....19
Historia....20
Del problema a la solución....21
De que trata este documento....22
¿Quien debe leer este documento?....22
Requisitos previos....22
Entendiendo el concepto....22
Principio de funcionamiento....22
CAPITULO 1: Preparando nuestro banco de trabajo....24
¿Que es una toolchain?....25
Toolchains estilo LFS....25
Toolchains estilo UTC (Ultra Thin Client)....25
El gran enemigo: el tamaño....26
La elección de la librería C standard para este escenario:....26
Nuestra navaja suiza: BUSYBOX....26
SVGALIB o como convertirse en X sin dejar nunca la consola:....27
SVNCVIEWER, la aplicación diseñada para nosotros:....27
CAPITULO 2: Comenzando el desarrollo....28
Verificando que distribución Gnu/Linux usar....29
Nomenclatura....29
Creando un nuevo usuario....30
Aplicaciones necesarias....31
El camino de la construcción....33
Preparando el nuevo KERNEL....33
Configurando el nuevo KERNEL....34
'CODE MATURITY LEVEL OPTIONS'....40
'LOADABLE MODULE SUPPORT'....42
'PROCESSOR TYPE AND FEATURES'....44
'GENERAL SETUP'....45
'BLOCK DEVICES'....48
'NETWORKING OPTIONS'....49
'ATA/IDE/MFM/RLL SUPPORT'....50
'NETWORK DEVICE SUPPORT'....52
'CHARACTER DEVICES'....55
'FILE SYSTEMS'....57
'CONSOLE DRIVERS'....58
Finalizando el nuevo KERNEL....59
Construyendo uClibC....59
Creando nuestro Root file system....62
Instalando uClibC en nuestro Root file system....64
Construyendo BUSYBOX....64
Comprobando la buena salud de BUSYBOX....65
CAPITULO 3: Implementando el acceso remoto gráfico....67
De la caja a la pantalla....68
Instalando las librerias matematicas en rfs....68
Configurando e instalando SVGAlib....69
Ajustando nuestro Makefile....69

Configuracion personalizada de SVGAlib (Makefile.cfg)....69
Configuracion e instalacion de svncviewer....71
Compilando svncviewer....71
Como funciona init en este sistema....73
Hacia la MICRODISTRO en diskette....74
Es tiempo de jugar....74
Metiendo todo en el diskette....75
Instalando el KERNEL y el disco RAM dentro del diskette....77
¡PERO TODAVIA HAY MAS!....78
CAPITULO 4: Implementando nuestro Servidor UTC....80
Implementando nuestro SERVIDOR VNC por demanda....81
¿Pero, por que tantos servicios?, ¿que hace cada uno?....81
Entendiendo Xwindow....81
Entendiendo TFTP....81
Entendiendo NFS....81
Entendiendo DHCP....82
Preparando nuestro SERVIDOR....83
Dos caminos, una eleccion....83
El camino facil....83
El camino dificil....84
¡Quiero mi implementacion ya!....85
Implementando un Servidor UTC en Fedora 10....85
Habilitando XDMCP....86
Comprobando la existencia de xinetd....88
Configurando xinetd & sus servicios - Instalando vnc-server....88
Configurando xinetd & sus servicios - Instalando vnc-server....88
¡Todavia hay mas!....90
¡Hemos terminado!....94
¡Rueda VNC!....94
Implementando un Servidor UTC en CentOS 4.* & 5.*....95
xinetd y vncserver....95
Buscando VNC....95
Configurando xinetd en CentOs 5.*....97
¡Rueda VNC!....100
Implementando un Servidor UTC en OpenSuse 10....101
Arranque y configuracion....101
Configuracion de la instalacion....101
Inicio de sesión....102
Instalando los servicios necesarios....102
Configurando nuestros servicios....102
DHCP-SERVER....102
Breve explicacion del archivo de configuracion dhcpd.conf....103
Parametros exclusivos para los clientes....104
TFTP Server....105
NFS Server....105
Configuracion de XDMCP....105
CAPITULO 5: Prescindiendo del diskette para UTC....107
Eliminando el Diskette o “como transportar Kernel & RFS por red”....108
Preparando nuestro Kernel....108
Convirtendo a nuestro Kernel en una imagen booteable....109

Habilitando el resto de los servicios....110
Servicio TFTP....110
Servicio NFS....111
Servicio DHCP....111
Palabras finales....114
ANEXOS....115
ANEXO 1 ¿Como funciona Etherboot?....116
ANEXO 2 ¿Que es DHCP?....118
ANEXO 3 ¿Que es TFTP?....119
ANEXO 3b TFTP, otra aproximacion....121
ANEXO 4 ¿Que es NFS?....122
NFS otra aproximacion....123
ANEXO 5 ¿Que es SVGAlib?....125
ANEXO 6 ¿Que es un disco RAM o RAMDISK?....126
ANEXO 7 Breve guia de referencia a vi....127
ANEXO 8 Virtual Network Computing o VNC ¿que es?....129
ANEXO 8b VNC, otra aproximacion....130
ANEXO 9 Etherboot en disco duro....132
ANEXO 9b EtherBoot en Disco duro serializado....134
ANEXO 10 Kernel cmdline options (comandos al vuelo para el Kernel, como funcionan)....135
ANEXO 11 Configurando XDMCP (X Desktop Manager Control Protocol)....136
ANEXO 12 Cosas a tener en cuenta antes de instalar LTSP....137
ANEXO 13 Implementacion de un entorno LTSP desde cero en Ubuntu 139
ANEXO 14 Open SuSe 10 y LTSP desde cero: 146
ANEXO 15 Entrada tipica de lts.conf (configuracion de clientes LTSP)....152
ANEXO 16 dhcpd.conf tipico para UTC/LTSP....157
ANEXO 17 /etc/exports tipico para UTC/LTSP....159
ANEXO 18 Como usar los clientes LTSP como pequeños servidores de impresion (Configuración lado cliente)....160
ANEXO 19 Como usar los clientes LTSP como pequeños servidores de impresion (Configuración lado Servidor)....161
ANEXO 20 LOCAL MEDIA....162
ANEXO 21 Mapa de teclado Ingles para svncviewer 164
ANEXO 21b Mapa de teclado Español para svncviewer 167
ANEXO 22 Tipos de chips de memoria no-volatiles....170
ANEXO 23 Lista de NICs (placas de red) soportadas por EtherBoot:....172
ANEXO 24 Proceso de arranque del Kernel Linux....177
ANEXO 25 Guía rápida para comprimir y descomprimir archivos....180
ANEXO 26 Mi .config del Kernel 2.4.22....182
ANEXO 27 Mi Makefile.cfg de svgalib....191
ANEXO 28 Mi .config de uClibC....196
ANEXO 29 Mi Config.h de busybox....198
ANEXO 30 Mis scripts para trabajar mas comodo....206
ANEXO 31 Troubleshooting....207
DHCP....207
Diagnostico, solucion de problemas (I)....207
Diagnostico, solucion de problemas (II)....207
TFTP....207
“File Not Found”....207
“nfs warning: server not responding...”....208

NFS a la hora de montar mi RootFileSystem, emite codigo de error “101”.....208
XDMCP.....209
Debuggeando “SYSLOG”.....209
Floppy imposible de leer.....209
Error Etherboot F?F?F?.....210
Etherboot descarga el Kernel, pero imprime los puntos muy lentamente (cada nn seg).....210
Problemas en la resolucion de pantalla (aka ¡Monitores Samsung SyncMater 3-3Ne no funcionan con LTSP!).....210
¿Renegando con la rueda del Mouse?.....210
El teclado se dispara! repite mas de 50 caracteres/seg!.....211
¡La terminal se freeza cuando navego por internet con firefox o uso openoffice!.....211
La terminal se freeza aleatoriamente.....211
Mensaje “syslogd: cannot write to remote file handle on 192.168.0.254:514”.....211
Kernel panic cuando trata de detectar una placa de red ISA!.....212
Mensaje “Creating new ramdisk to hold our root fs...”.....212
Mensaje “make: *** [init/main.o] Error 1”.....213
Códigos de error de LiLo.....213
No Init Found.....214
Cantidad de errores de compilacion.....214
Probemas clasicos de memoria una vez que arranca el Kernel.....215

Introducción

Descripción

Este documento intenta cubrir un nicho en donde no he encontrado una solución real al problema que se me presenta desde hace 5 años a la fecha.
Proveer una solución a medida con respecto al refuncionalizado de antiguas maquinas pertenecientes a la arquitectura i486, con un mínimo disponible de memoria RAM base de 4MB, sin discos o lectoras de ningún tipo, el cual me ofrezca un ambiente de trabajo con GUI (Graphical User Interface), acceso y navegación por Internet y posibilidad de uso de aplicaciones tales como paquetes ofimáticos, aplicaciones educativas, juegos didácticos, visualización en alta resolución (miles de colores) y demás necesidades que se presentan hoy en día a la hora de usar una computadora.

Historia

El protagonista de este escenario, es la situación en que se encuentra el Ministerio de Educación de la Provincia de Córdoba en el año 2004, el cual cuenta con un parque tecnológico obsoleto de mas de 1500 computadoras, entre ellas i386, i486 y series Pentium (I, II) repartido en mas de 70 escuelas primarias que formaron parte de un programa educativo a principio de los años 90 llamado 'PROINTEC', el cual consistía en un aula informática con 19 computadoras IBM PS/ValuePoint SX25Mhz con 4MB de RAM y un SERVER IBM PS/2 DX33 con 8MB RAM, todas interconectadas en modo anillo con cable rg59 (coaxial) 10Base2, en las cuales corría una red Novell 3.1* + Windows 3.11 (para trabajo en grupos).

El problema surge cuando la Ex-Gerencia de Sistemas (actual Dirección de Sistemas) del Ministerio de Educación de la Provincia. De Córdoba, en Febrero de 2004 me consulta sobre la posibilidad de reflotar este parque, ya que los fondos son escasos para cubrir con nuevo hardware la totalidad de las Escuelas con esta carencia en toda la región de la Provincia.
Adicionalmente a esto, el Ministerio de Educación cuenta en la actualidad con cerca de 1500 computadoras Celeron 556Mhz con 64Mb RAM en mas de 300 escuelas también primarias.

Del problema a la solución

Obviamente la solución viene de la mano de GNU/LINUX, ya que este es open source (código abierto), libre, portable a cualquier plataforma y arquitectura, esta estrechamente ligado al manejo en modo usuario del hardware y es totalmente maleable (modular o monolítico), como veremos en breve.

Ademas la solución debe contemplar el uso de una Interface gráfica de alta resolución para el usuario, en la cual este pueda navegar por Internet, utilizar paquetes ofimáticos y cualquier otra aplicación que crea conveniente.
El desafío en si es poder reutilizar este hardware de legado con su configuración por defecto (4MB RAM) como base.

En un principio se logro arrancar estas maquinas con 8MB RAM, pero esto implicaba sacrificar 2 maquinas por una, ya que el sistema de paridad de las memorias es propietario de IBM y solo aceptan a su lado otra memoria IBM.
Esta solución fue presentada por Pablo Perez de Angelis, en el tiempo que prestaba servicios en la ex-Gerencia de Sistemas del Ministerio de Educación de la Provincia de Córdoba – Argentina.

Actualmente he re-implementado la solución basada en 8MB RAM para bajar sus requerimientos a tan solo 4MB y así aprovechar el hardware en su totalidad.

Esta solución consta de 3 versiones, las cuales difieren entre si en varios aspectos:

•Versión N°1: Disquete de arranque conteniendo Kernel + Root File System + cliente vnc en un mínimo de memoria física de 8MB, implementado por Pablo Perez de Angelis (Ex Gerencia de Sistemas – Ministerio de Educación de la Provincia de Córdoba – Argentina). Año 2005.
•Versión N°2: Disquete de arranque conteniendo Kernel + Root File System + cliente vnc en un mínimo de memoria física de 3.8MB, implementado por Alberto Castillo. Abril 2009.
•Versión N°3: implementación basada en etherboot (ROM NIC) + tftp (para cargar el Kernel) + NFS (para montar el Root File System) + cliente vnc en un mínimo de memoria física de 3MB implementado por Alberto Castillo. Mayo 2009.

En este documento voy a centrarme en las dos ultimas versiones, ya que la primera se basa íntegramente en la solución creada por el Húngaro Csaba Henk, en sus documentos:

“Making a UclibC-based bootfloppy” y *“Setting up a graphical thin client using the svga vncviewer”*.

En su proyecto LOWLIFE:
(http://www.math-inst.hu/~ekho/lowlife/)

y en los hints de LFS:
http://www.linuxfromscratch.org/hints/downloads/files/uclibc-bootfloppy.txt

http://www.linuxfromscratch.org/hints/downloads/files/svnc-thinclient.txt

Excepto por la inclusión del mapeo del teclado en español y el arranque automático del cliente VNC una vez cargado el sistema, que Pablo Perez de Angelis incluye en la misma.

Las restantes versiones también se basan en estos documentos de Csaba Henk e incluyen el mapeo de teclado en español, pero por condiciones obvias al escenario presentado, se deben realizar distintas modificaciones para poder lograr un sistema a medida con la escasez de recursos a nivel hardware que aquí se presenta.

De que trata este documento

Trata del desarrollo e implementación paso a paso de una micro distribución GNU/LINUX especifica para el uso de computadoras i486 como clientes finos, en dos variantes: disquete de arranque conteniendo toda la microdistribución, entregando una pantalla de LOGIN gráfico en modo remoto para su uso y la implementación de esta micro distribución para que pueda ser transportada mediante una red física de computadoras hacia este mismo cliente fino, sea montada y presente también una pantalla gráfica de LOGIN.

¿Quien debe leer este documento?

Las personas interesadas en comprender como funciona por dentro un sistema básico tipo GNU/LINUX. Las personas que no se amedrentan ante el primer mensaje de error en pantalla. Las personas que les gusta tomar el camino mas dificil de las cosas. Las personas con ganas de aprender un poco mas.

Requisitos previos

Poseer conocimientos básicos de la jerarquía de archivos y directorios de Linux, uso básico de bash, conocimiento en redes TCP/IP, conocimientos básicos de compilación en Linux

Entendiendo el concepto

Aquí lo que planteo, ademas de poder aprovechar una maquina obsoleta para poder volver a hacerla funcional según las necesidades actuales, es el hecho de que una maquina de esta arquitectura Intel 486 como cliente fino, implementada en una red de computadoras, me trae la ventaja de que esta diseñada para ser silenciosa, o sea que no tiene partes mecánicas móviles, al no tener un cooler en el microprocesador para enfriarlo, no tiene lectora de CD con sus respectivos motores y lectora láser, no posee disco duro con sus brazos móviles (excepto si lo usamos como gestor de arranque etherboot) y en la versión N°3 de este mismo sistema eliminamos también el uso de la disquetera con su motor y su cabeza lectora magnética, logrando así un hardware que ahorra energía eléctrica hasta un 60% con respecto de un sistema medio tipo standalone que requiera cumplir las mismas funciones, por lo tanto es ecológico Lo único móvil apreciable es el ventilador de la fuente de alimentación incorporada al chasis de la misma.

Por otro lado lo que queremos es un sistema gráfico atractivo para que sirva como enganche y sea de extrema utilidad a los niños de las escuelas iniciales y primarias. Ademas también necesitamos un compendio de aplicaciones tanto de oficina como educativas y recreativas para hacer realmente eficiente este proyecto.

Principio de funcionamiento

Básicamente, el principio es muy simple, necesitamos un pequeño Kernel Linux que pueda cargarse en memoria, para luego montar todo su Root file System también en memoria (RAMDISK), ejecutar su propio 'init' (el padre de todos sus procesos), el cual ejecutara un cliente VNC que se conectara remotamente a un servidor gráfico, resultando en una pantalla de bienvenida o LOGIN.

El concepto tiene 2 variantes, la primera es que usamos un disquete de 1.44MB como dispositivo de arranque, en el cual tenemos al principio del mismo, el Kernel, el cual es parchado para que una vez cargado en memoria sepa en que bloque físico del disquete se encuentra todo su file System, reserve memoria y lo monte en ella, a esto se llama RAMDISK.

En la otra variante, utilizamos una rutina de arranque alojada en una ROM, la cual físicamente se encuentra conectada a la placa de red del cliente y hace una petición de identidad mediante el protocolo dhcp (Dynamic Host Configuration Protocol), un disquete con esa misma rutina de arranque o un disco duro, donde en sus primeros bloques se alojara esta misma rutina, esto en otras palabras significa que mediante esta rutina de arranque, mediante el protocolo dhcp, el cliente pide una dirección IP para poder ahora traer el Kernel, el cual es transferido mediante otro protocolo, el tftp (Trivial File Transfer Protocol).

Una vez que el Kernel se ha desempaquetado y montado en memoria, pide al protocolo nativo de Unix/Linux NFS (Network File System) que le envíe el Root file System completo y que lo monte en memoria.

A partir de aquí, una vez que es montado, el proceso es idéntico a lo anteriormente descripto, se ejecuta init, luego éste llama a un cliente VNC, el cual se conecta a nuestro servidor y nos ofrece una pantalla gráfica de LOGIN.

Una cosa a tener en cuenta es que necesitamos que todo ocupe menos de 1.44MB para poder utilizar un disquete de arranque, para eso necesitamos de herramientas diferentes a las proporcionadas por una distribución Linux standard.

CAPITULO 1: Preparando nuestro banco de trabajo

Preparando el entorno de trabajo

Para crear nuestro sistema operativo, primero necesitamos entender el concepto de las toolchains.

¿Que es una toolchain?

Una toolchain es un conjunto de herramientas que nos ayudaran en nuestra tarea de compilación y construcción de nuestro micro sistema Linux en su totalidad, tanto el Kernel como las aplicaciones que se correrán en nuestro file System

Hay dos formas de tener a punto nuestra toolchain para este trabajo tan especifico, ya que recordemos, estamos por trabajar sobre la arquitectura i386/i486 exclusivamente.
La primera es al estilo LFS (Linux From Scratch) y la segunda al estilo UTC.

Toolchains estilo LFS

Los desarrolladores de LFS tienen un método infalible para tener un entorno de trabajo totalmente independiente de la maquina Host (la maquina en la cual nos sentamos a implementar esta microdistribución).

Ellos básicamente, trabajan en una partición del disco duro solo dedicada a esto, construyen todo un set de herramientas, tales como un compilador gcc, la librería C, los enlazadores dinámicos, las binutils, para ser usados de forma temporal para poder construir ahora si, una nueva toolchain para poder compilar la nueva librería C que elijamos, el Kernel, las aplicaciones,etc.

De este forma necesitamos construir 2 (dos) toolchains para nuestra implementación, una temporal que aloje a la que sera definitiva para construir nuestro sistema.

Dado que aquí pretendemos construir un sistema realmente minimalista, es mas, el concepto va mas allá de minimista para convertirse en una aproximación a un sistema embebido o incrustado, ya que tiene exclusivamente las funcionalidades necesarias para un especifico funcionamiento y nada mas, en mi opinión personal me parece mucho esfuerzo en tiempo y trabajo este método, pero si necesitamos realmente construir un sistema operativo Linux completo y totalmente optimizado, esta también para mi opinión, es la mejor forma de lograrlo.

Toolchains estilo UTC (Ultra Thin Client)

Bueno, esto simplemente responde a una necesidad de tiempo y trabajo.
Este método simplemente se refiere a trabajar en una maquina Host (en la cual vamos a armar nuestro nuevo sistema) que no sea mas grande que una Pentium 3 con 256MB RAM como máximo y que contenga una instalación de un viejo sistema operativo Linux, el cual posea nativamente antiguas herramientas de compilación perfectamente interrelacionadas entre si, para evitar la construcción de una toolchain temporal.

Necesitamos prepararla con todas sus herramientas de desarrollo (cuando instalamos podemos ver categorías, buscar desarrollo) e instalarle un Server ssh, ya que la forma mas cómoda es trabajar remotamente desde nuestra moderna maquina con CentOs 5, Fedora 12, Ubuntu 10 o lo que sea que estén usando actualmente de cara a esta maquina modesta que hará todo el trabajo de compilación y construcción

Recomiendo Red Hat 9 o Mandrake 10, ya que estas distribuciones poseen principalmente gcc-3.3.2, o cualquier distribución que la posea, ya que con ella se pudo realizar este desarrollo con éxito

El gran enemigo: el tamaño

El problema de una implementación de esta envergadura es que físicamente estamos confinados a trabajar con un tope de memoria de 4MB como máximo

De esta forma necesitaremos utilizar tanto una librería como un conjunto de aplicaciones que han sido (y aun siguen siéndolo) desarrolladas pensando en el tamaño como factor principal.

La elección de la librería C standard para este escenario:

La respuesta inmediata de cualquier persona que ha estado involucrada en sistemas Linux, siquiera en una mínima compilación seria: glibc.

Como dice Csaba, la ventaja de glibc es que la tenemos a mano, viene como parte de cualquier distribución Linux, pero su desventaja es que su tamaño es fenomenalmente enorme para este tipo de implementaciones.

Aquí es cuando aparece ante nosotros una librería que esta diseñada específicamente para estos casos: uClibC, esta cumple con los estándares y es ampliamente usada en desarrollos de sistemas embebidos a nivel mundial.
Para aclarar un poco mas las cosas esta es una tabla que nos muestra la comparación entre las dos librerías C:

Aplicación en C	**Compilada con librerías compartidas**		**Compilado estáticamente**	
	glibc	**uClibC**	**glibc**	**uClibC**
Texto “Hola Mundo”	4.6KB	4.4KB	475KB	25KB
BusyBox	245KB	231KB	843KB	311KB

Tabla adquirida de www.free-electrons.com

Nuestra navaja suiza: BUSYBOX

Esta aplicación es una suerte (como dicen sus desarrolladores) de navaja suiza todo terreno para implementación de distribuciones Linux embebidas.
Dentro de su código esta aplicación alberga una formidable cantidad de comandos que son utilizados a diario por cualquier usuario Linux, desde 'ls' hasta 'grep' pasando por 'ifconfig' y 'ssh'.

Básicamente es un ejecutable que posee enlaces de comandos a si mismo, el cual consulta en un indice interno y ejecuta código correspondiente al comando solicitado, por ejemplo si ejecutamos 'ls' en nuestro back-end construido con BusyBox, este desplegara la información de nuestro directorio de trabajo como si se tratara del 'ls' nativo de nuestra distribución Linux preferida.

SVGALIB o como convertirse en X sin dejar nunca la consola:

Como dice WikiPedia en español:
“SVGAlib es una librería gráfica de bajo nivel y de código abierto para Linux y FreeBSD que permite que los programas cambien el modo de vídeo y muestren gráficos a pantalla completa. Algunos juegos populares como Doom la utilizaron en sus lanzamientos originales.”

De esta forma develo el misterio de la pregunta: “como es posible tener un entorno gráfico GNOME en una i486 con solo 4MB de RAM?”, pero sola de por si no nos sirve de nada esta librería, para eso necesitamos una aplicación que pueda conectarse a nuestro servidor y ofrecernos una pantalla de LOGIN gráfico.

SVNCVIEWER, la aplicación diseñada para nosotros:

Como dice linux.about.com:
“Cliente VNC para SVGAlib, el cual nos muestra un escritorio remoto, independientemente de la plataforma que usemos, el cual se conecta a un servidor VNC desde una consola de texto”.

CAPITULO 2: Comenzando el desarrollo

Verificando que distribución Gnu/Linux usar

Antes que nada verifiquemos que tenemos instaladas en nuestra distribución (recuerden: recomiendo Red Hat 9 o Mandrake 10) las herramientas de compilación

También recomiendo una instalación desde cero e incluirle las correspondientes herramientas.
El entorno ideal de trabajo, es a mi punto de vista, usar nuestra maquina con su distribución moderna de Linux como consola remota hacia la distribución que construirá nuestra microdistribución, mediante un switch y 2 patch-cores (2 cables de red UTP CAT 5), mediante ssh.

Por razones obvias, no me voy a detener a explicar el funcionamiento de ssh ni la implementación de una red domestica de dos computadoras.

Nomenclatura

Quiero dejar algunas normas para evitar escribir de mas y que queden de ahora en mas asimiladas a lo largo de toda la documentación:

Cuando encuentres el símbolo '#' (sin las comillas simples) quiero decir que debemos trabajar como SUPER USUARIO o root. A no ser que se especifique otro usuario.

Cuando encuentren un símbolo '$' (sin las comillas simples) quiero decir que debemos trabajar como un usuario restringido, en este documento, asumimos que este usuario restringido es el que vamos a usar para construir la microdistribución A no ser que se especifique otro usuario.

Cuando observes en un comando la barra invertida '\' (sin las comillas simples), significa que debes presionar la tecla 'Enter' y continuar debajo sin agregar ningún espacio al resto de los argumentos.

No borrar ningún directorio (especialmente las fuentes desempaquetadas) o archivo, a menos que se especifique lo contrario.

El comando 'su' sirve para cambiar de usuario al vuelo. Si no damos ningún argumento, bash interpreta que queremos ser superusuarios. La forma mas común es usar a su de la siguiente manera: 'su -', el cual indica al interprete que no herede nada del usuario anterior. Pero cuidado, en este documento, vamos a usar de forma premeditada el uso de superusuario heredando las características del usuario anterior, en este caso el usuario restringido. Así que no es lo mismo 'su' que 'su -' y cuando indico cambiarte a modo superusuario con 'su', pretendo que escribas 'su' y no 'su -'.

Cuando uses 'vim' (mi editor de textos favorito), la forma de guardar los cambios y salir es presionando la tecla escape, luego la tecla ':' seguida las teclas 'w' y 'q'. Así de esta forma cuando indico:
'(esc):wq', estoy ejecutando los pasos que acabo de describir.
También significa que '()' indica que debemos tipear el contenido encerrado entre paréntesis en el editor vim.

Por favor, no te confundas, las comillas simples las uso para enfatizar solo en el texto explicativo. Cuando la veas en los ejemplos de ingreso de código, agregalas.

Creando un nuevo usuario

Para trabajar cómodos y de la forma mas independiente posible, vamos a crear un nuevo usuario en nuestro sistema recién instalado, donde trabajaremos de ahora en mas. El nuevo usuario se llamara 'utc' (alias de 'ultra thin client'):

```
# adduser utc
# passwd utc
```

Con esto hemos creado nuestro usuario 'utc' el cual sera el encargado de la construcción del nuevo sistema.

Ahora ingresemos como este nuevo usuario:

```
# su - utc
```

****NOTA: el símbolo menos (-) le indica a 'su' que debe cargar el entorno de usuario que se la ha asignado por defecto y no heredar el entorno del usuario anterior.****

```
$ pwd
/home/utc
```

Vemos que estamos dentro de nuestro directorio de trabajo. Ahora, lo primero que vamos a hacer es modificar nuestro entorno de trabajo para que responda a nuestras necesidades.

Ejecutamos:

```
$ vim .bash_profile
```

Luego agregamos:

```
export CFLAGS=-Os
export CC=i386-uclibc-gcc
```

(esc):wq

Estas dos variables de entorno, le especificaran a nuestro sistema que a la hora de compilar utilice la herramienta de optimización de tamaño de gcc -Os y que por defecto cuando llegue el momento de usar un compilador C, utilice el compilador de uClibC como herramienta standard de compilación

Chequeamos que la versión de gcc sea igual a 3.3.2. En Mandrake 10 por ejemplo, tenemos esta respuesta:

```
$ gcc -v
Reading specs from /usr/lib/gcc-lib/i586-mandrake-linux-
gnu/3.3.2/specs
Configured with: ../configure --prefix=/usr --libdir=/usr/lib
--with-slibdir=/lib --mandir=/usr/share/man
--infodir=/usr/share/info --enable-shared --enable-threads=posix
--disable-checking --enable-long-long --enable-__cxa_atexit
```

```
--enable-clocale=gnu --enable-languages=c,c+
+,ada,f77,objc,java,pascal --host=i586-mandrake-linux-gnu --with-
system-zlib
Thread model: posix
***gcc versión 3.3.2 (Mandrake Linux 10.0 3.3.2-6mdk) ***
```

A continuación creamos un directorio donde guardaremos las aplicaciones necesarias y luego nos posicionaremos dentro:

```
$ mkdir sources
$ cd sources
```

Aplicaciones necesarias

Recomiendo encarecidamente usar estas versiones de las aplicaciones, ya que han sido testeadas con éxito en este desarrollo.

Linux 2.4.22
http://www.kernel.org/pub/Linux/Kernel/v2.4/Linux-2.4.22.tar.bz2

uClibC 0.9.19
http://uclibc.org/downloads/old-releases/uClibc-0.9.19.tar.bz2

BusyBox 0.60.5
http://busybox.net/downloads/legacy/BusyBox-0.60.5.tar.bz2

tightvnc 1.2.8

****NOTA: lamentablemente las fuentes de esta versión están extintas, solo encontramos en el site oficial del grupo que desarrolla tightvnc fuentes a partir de la versión 1.2.9, pero a no preocuparse, ya que yo proveo ademas del documento todas las fuentes, en donde encontraremos la versión correcta de tightvnc, de todas formas si te sientes animado y crees que la suerte esta de tu lado para compilar la versión inmediatamente superior (1.2.9), acá te dejo el link:****

http://sourceforge.net/project/downloading.php?group_id=14067&filename=tightvnc-1.2.9_unixsrc.tar.bz2&a=81146102

SVGAlib 1.9.18
http://my.arava.co.il/~matan/svgalib/svgalib-1.9.18.tar.gz

svncviewer 0.1
http://6a.edskes.com/opsys/Linux/sunsite/X11/desktop/svncviewer-0.1.1.tar.gz

****NOTA: me costo muchísimo trabajo encontrar las fuentes de este programa que ha quedado perdido en el tiempo y actualmente rechazado por todas las distribuciones al considerarlo obsoleto.*

La ironía es que este es el único software desarrollado para aceptar conexiones VNC remotas desde una consola en modo texto, una lastima que el responsable del proyecto lo haya abandonado y que nadie haya querido seguir adelante, aunque también reconozco que la función que debe cumplir la

cumple y muy bien, excepto por pequeños detalles quizás de índole estética, tal como la falta de encendido del led de la tecla BloqNum del teclado numérico, ya que el mismo se activa pero el led testigo no.
Tengo la esperanza de que con esta implementación, resucite esta aplicación única en su tipo.

Ahora descargamos y descomprimimos los archivos en el nivel superior (~):

Descargamos:

```
$ wget -c http://www.kernel.org/pub/Linux/Kernel/v2.4/Linux-
2.4.22.tar.bz2
$ wget -c http://uclibc.org/downloads/old-releases/uClibc-
0.9.19.tar.bz2
$ wget -c http://busybox.net/downloads/legacy/BusyBox-
0.60.5.tar.bz2
$ wget -c
http://sourceforge.net/project/downloading.phpgroup_id=14067&filen
ame=\     tightvnc-1.2.9_unixsrc.tar.bz2&a=81146102
$ wget -c http://my.arava.co.il/~matan/svgalib/svgalib-
1.9.18.tar.gz
$ wget – c
http://6a.edskes.com/opsys/Linux/sunsite/X11/desktop/svncviewer-
0.1.1.tar.gz
$ wget – c
http://www.linuxfromscratch.org/patches/downloads/svgalib/svgalib-
1.9.18-\  demos_Makefile-1.patch
```

Descomprimimos:

```
$ tar jvxf  Linux-2.4.22.tar.bz2 -C /home/bootdisk
$ tar jvxf  uClibC-0.9.19.tar.bz2  -C /home/bootdisk
$ tar jvxf  BusyBox-0.60.5.tar.bz2 -C /home/bootdisk
$ tar zvf svgalib-1.9.18.tar.gz -C /home/bootdisk
```

****NOTA: No se si hace falta recordar que con las flechas de cursor y la tecla TAB podemos simplificar este trabajo.****

Volvemos a nuestro directorio inicial de trabajo:

```
$ cd
```

Ahora, antes de comenzar con la compilación de cada paquete, necesitamos ajustar el futuro acceso que tendrá uClibC al Kernel, al momento de su compilación

****NOTA: Este ajuste tuve que realizarlo, ya que si le indicaba un path absoluto o distinto de la palabra 'Linux' me era imposible compilar uClibC (increíble por cierto...).****

```
$ ln -s Linux-2.4.22 Linux
```

El camino de la construcción

El camino que vamos a tomar es el siguiente: primero, compilamos nuestro Kernel, para poder así dejar listas sus cabeceras, que serán las que a la hora de compilar uClibC, sean encontradas por esta librería, luego compilamos uClibC, para que sea la encargada de compilar nuestras aplicaciones de forma optimizada en tamaño, una vez compilada nuestra nueva librería, le toca a BusyBox ser compilado, una vez que ha sido también compilado, nos disponemos a crear nuestro árbol de directorios o RootFileSystem, el cual albergara al resto de las librerías y aplicaciones. Una vez listo nuestro RFS, es hora de compilar SVGAlib, por ultimo, compilamos svncviewer.
Una vez compilado todo, solo tenemos que ajustar nuestro nuevo entorno y decidir que variante vamos a utilizar: Disquete o Root over NFS.

Preparando el nuevo KERNEL

Entramos a nuestro Kernel:

```
$ cd Linux
```

El principal motivo de porque usamos un Kernel de la serie 2.4.* es porque en esta serie todavía se permite el booteo del Kernel sin ningún gestor de arranque, cosa que no es posible en la serie 2.6.*.

Esta es la principal ventaja que obtenemos al usar un disquete como método de arranque, ya que simplemente le decimos al Kernel que debe arrancar y luego fijarse en un bloque físico del mismo para poder cargar nuestro RFS.

Antes que nada, debemos dejar limpio el entorno de compilación de nuestro Kernel:

```
$ make clean && make mrproper
```

Con esto nos aseguramos que el directorio esta limpio.
La idea es compilar un Kernel que sea capaz de interactuar en red y que este optimizado al máximo en tamaño.

Para esto, le he seguido la pista a la configuración del Kernel propuesta por Csaba Henk (http://www.creo.hu/~csaba/lowlife/Linux-2.4.22/.config) , pero he ido un poco mas allá, llegando a obtener un Kernel de 509KB en modo bzImage.

Ahora vamos a usar la herramienta de optimización en tamaño postulada por Csaba Henk:

```
$ sed -e 's%-O2%-Os%g' -e '/^CFLAGS_KERNEL/s%\(^.*$\)%\1 -Os%'
Makefile > \
 Makefile.tmp &&
mv Makefile.tmp Makefile
```

bash nos va a preguntar si deseamos sobrescribir nuestro Makefile, le decimos que si.

De todas formas si no tienes ganas de escribir mucho, puedes directamente abrir el Makefile, localizar la linea #40 y reemplazar el contenido de CFLAGS_KERNEL por -Os.

*NOTA:*** Os es una letra 'O' seguido de la letra 's' no confundir con 0s (0 es un cero).*

Configurando el nuevo KERNEL

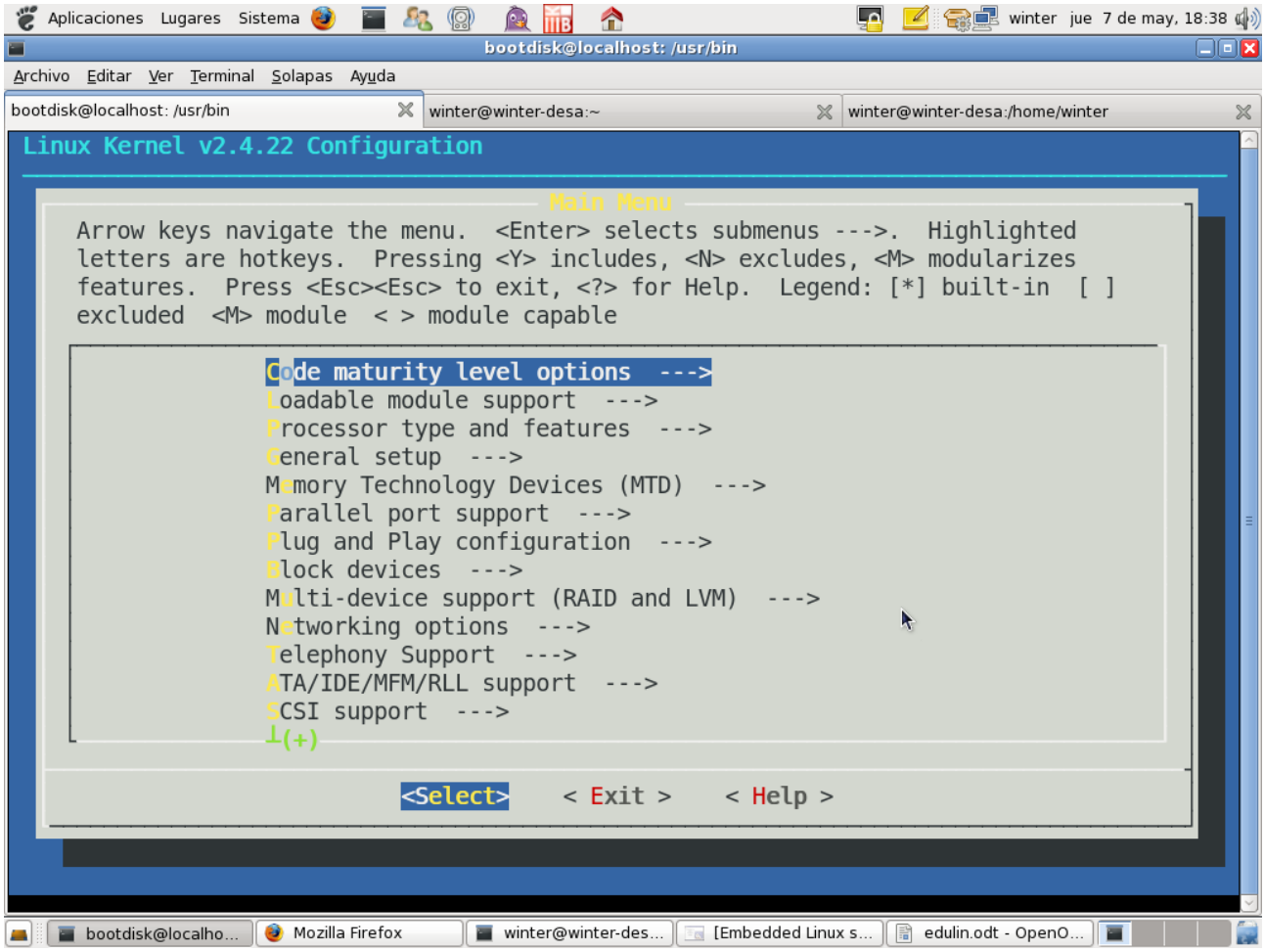
El método mas fácil es dejando desactivadas todas las opciones de construcción del Kernel:

```
$ make allnoconfig
```

Una vez hecho esto, entramos a la configuración gráfica:

```
$ make menuconfig
```

Obtenemos una pantalla como esta:



Usamos para agregar funcionalidades a nuestro Kernel la siguiente nomenclatura:

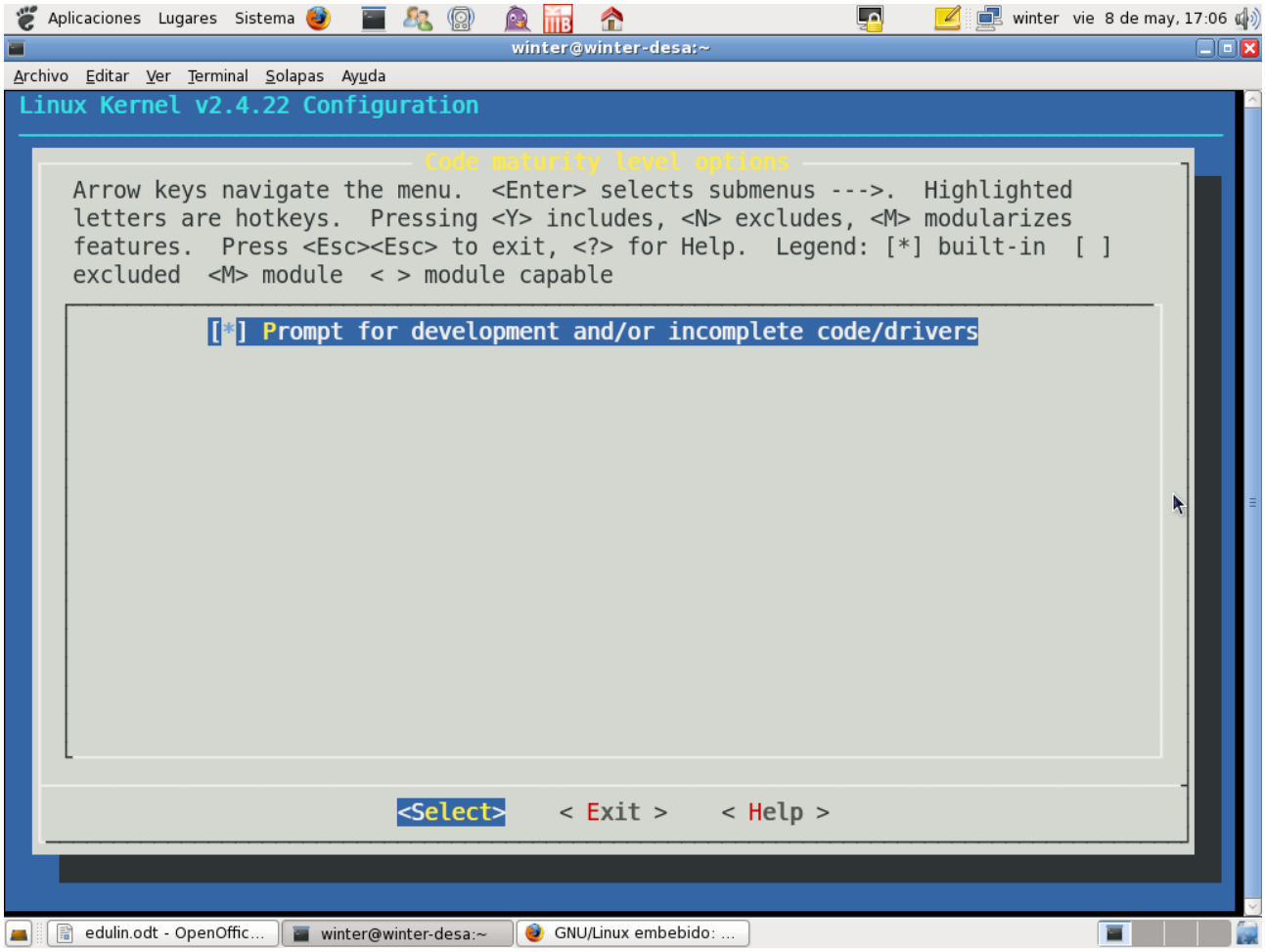
•[*] esta funcionalidad o característica se incluirá estáticamente dentro del núcleo
•<M> esta funcionalidad o característica se incluirá como modulo, el cual se cargara cuando el núcleo así lo crea conveniente.
•<*> significa que puede ser incluida la nueva característica estéticamente, pero también acepta su inclusión en forma de modulo.
•[ ] esta funcionalidad o característica NO se incluirá dentro del núcleo

Para cambiar entre estos tres estados, usamos la barra espaciadora, la cual nos mostrara sucesivamente cada uno.

Para navegar dentro de cada menú, usamos las teclas del cursor y la barra espaciadora.
La lista de las opciones por defecto que he dejado es la siguiente (opción por opción, paso a paso):

'CODE MATURITY LEVEL OPTIONS'

[*]Prompt for development and/or incomplete code/drivers



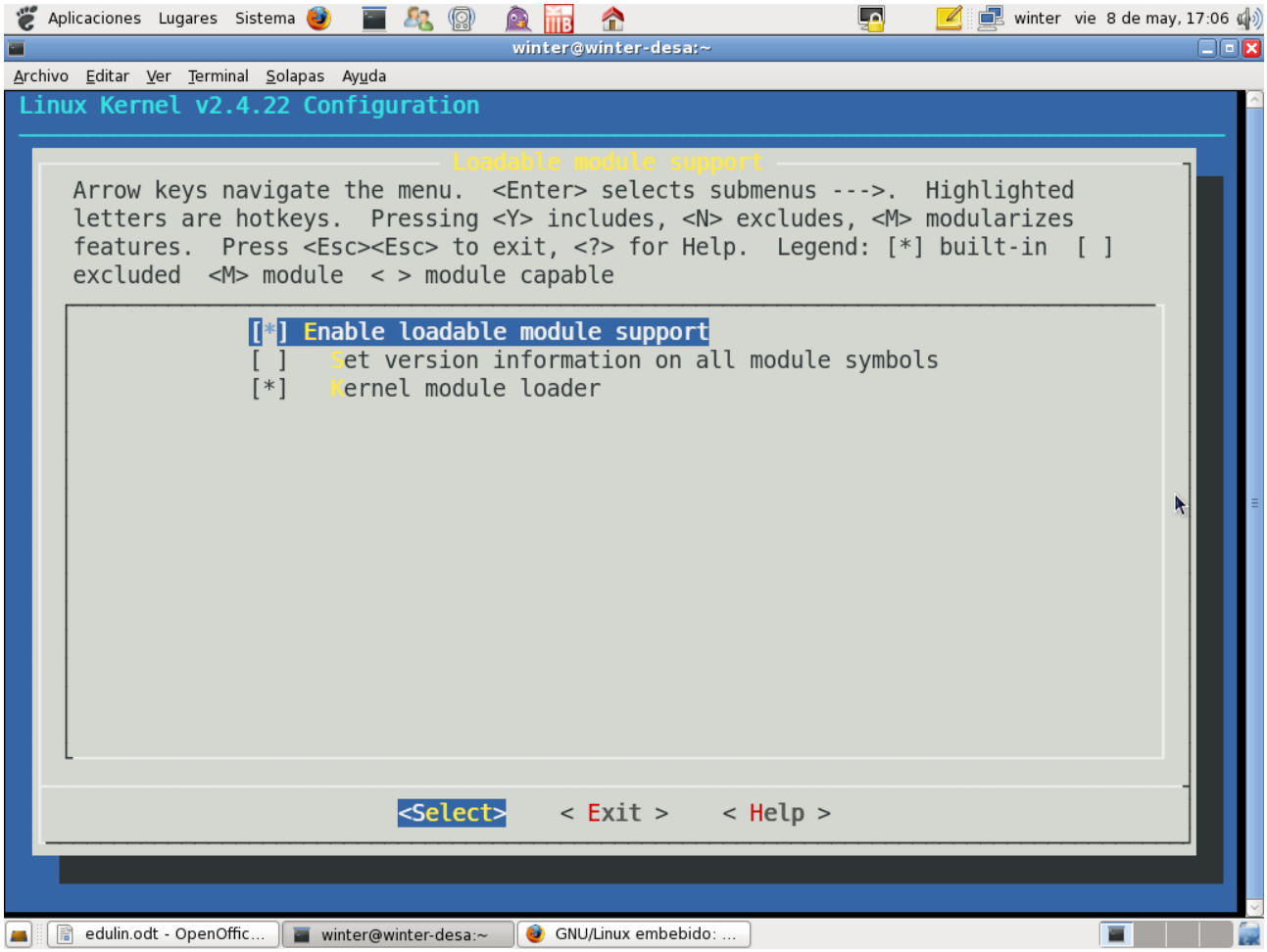
CONFIG_EXPERIMENTAL - Según la documentación oficial:
"Muchas de las cosas que Linux soporta, actualmente se encuentran en estado de desarrollo, donde la funcionalidad, estabilidad o nivel de desarrollo todavía no se encuentra maduro para el uso general. Esto es conocido como "tests – alfa" entre los desarrolladores". Habilitando esto, permitimos tener soporte para este tipo de drivers".

'LOADABLE MODULE SUPPORT'

[*] Enable loadable module support

[*] Kernel module loader

Por razones estrictamente técnicas, no usaremos ningún modulo, pero si dejaremos la opción de que el Kernel pueda cargarlos al vuelo, ya que esta funcionalidad la vamos a usar cuando ejecutemos las librerías gráficas.

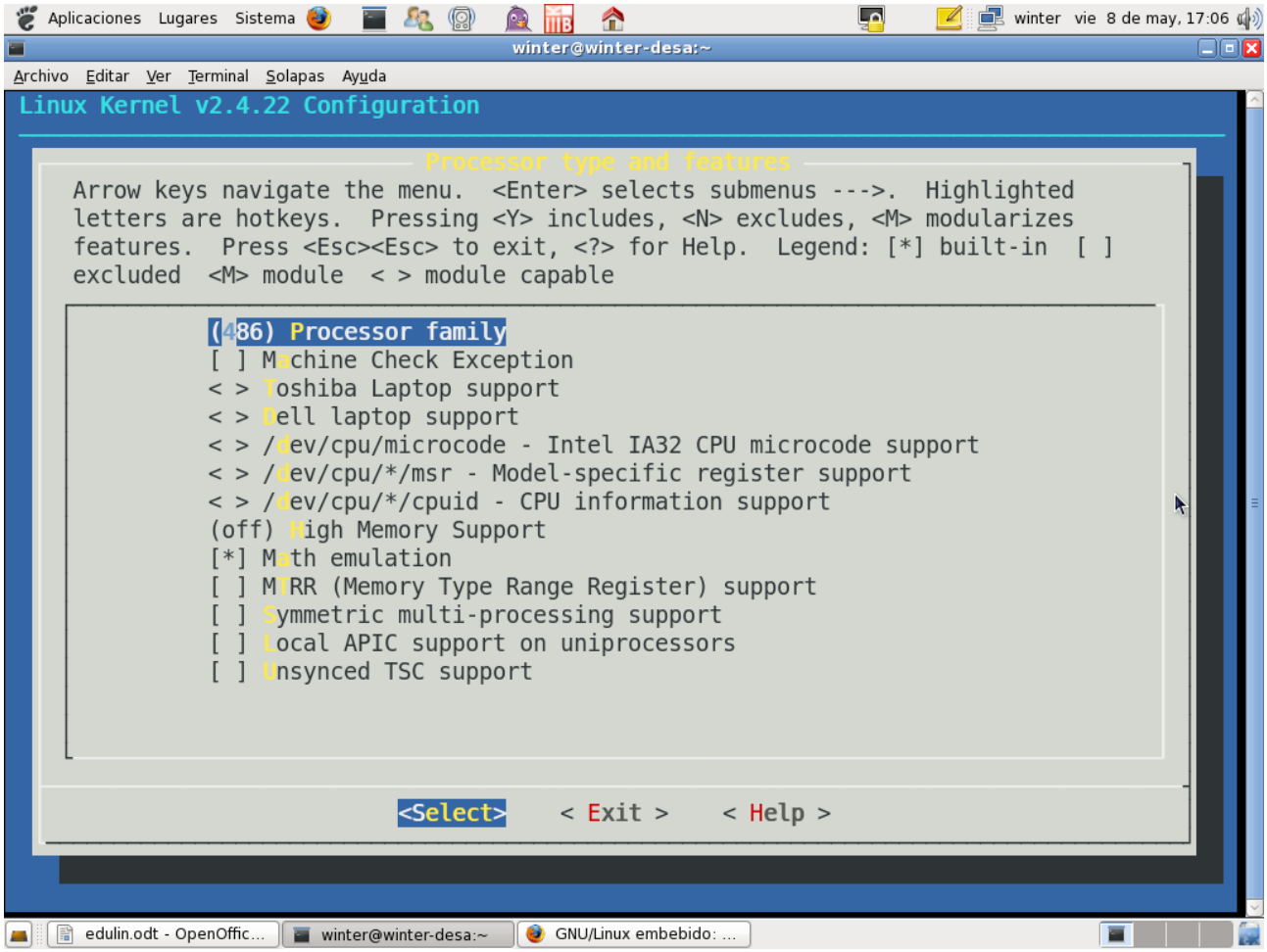


CONFIG_MODULES - Según la documentación oficial: *"Los módulos del Kernel son pequeñas piezas de código compilado, el cual puede ser insertado o removido del Kernel en plena ejecución, usando los programas 'insmod' y 'rmmod'. Para obtener esta características, debemos decirle al núcleo en su proceso de configuración, que compile los módulos con 'make modules'. Los módulos pueden ser tanto drivers de dispositivos, como sistemas de archivos, formatos de binarios ejecutables, y mas."*

CONFIG_KMOD - Según la documentación oficial: *"Normalmente cuando se han elegido algunos drivers o sistemas de archivos para ser creados como módulos cargables al vuelo, se tiene también la responsabilidad de cargar sus correspondientes módulos (usando los programas 'insmod' o 'modprobe') antes de que puedan ser usados."*

'PROCESSOR TYPE AND FEATURES'

(486) Processor family
(off) High Memory Support
[*] Math emulation



CONFIG_M386 - Según la documentación oficial: "*Este el el tipo de microprocesador de la CPU destino. Esta marca se usa para propósitos de optimización Si se desea correr el Kernel en la totalidad de las arquitecturas x86, debe elegirse '386' aquí De esta forma, el Kernel no necesariamente puede correr en arquitecturas mas antiguas de las que aquí se haya elegido. Por ejemplo, un Kernel optimizado para Pentium podrá correr en un PentiumPro, pero no necesariamente en un 486... (mas información en <Help>)*".

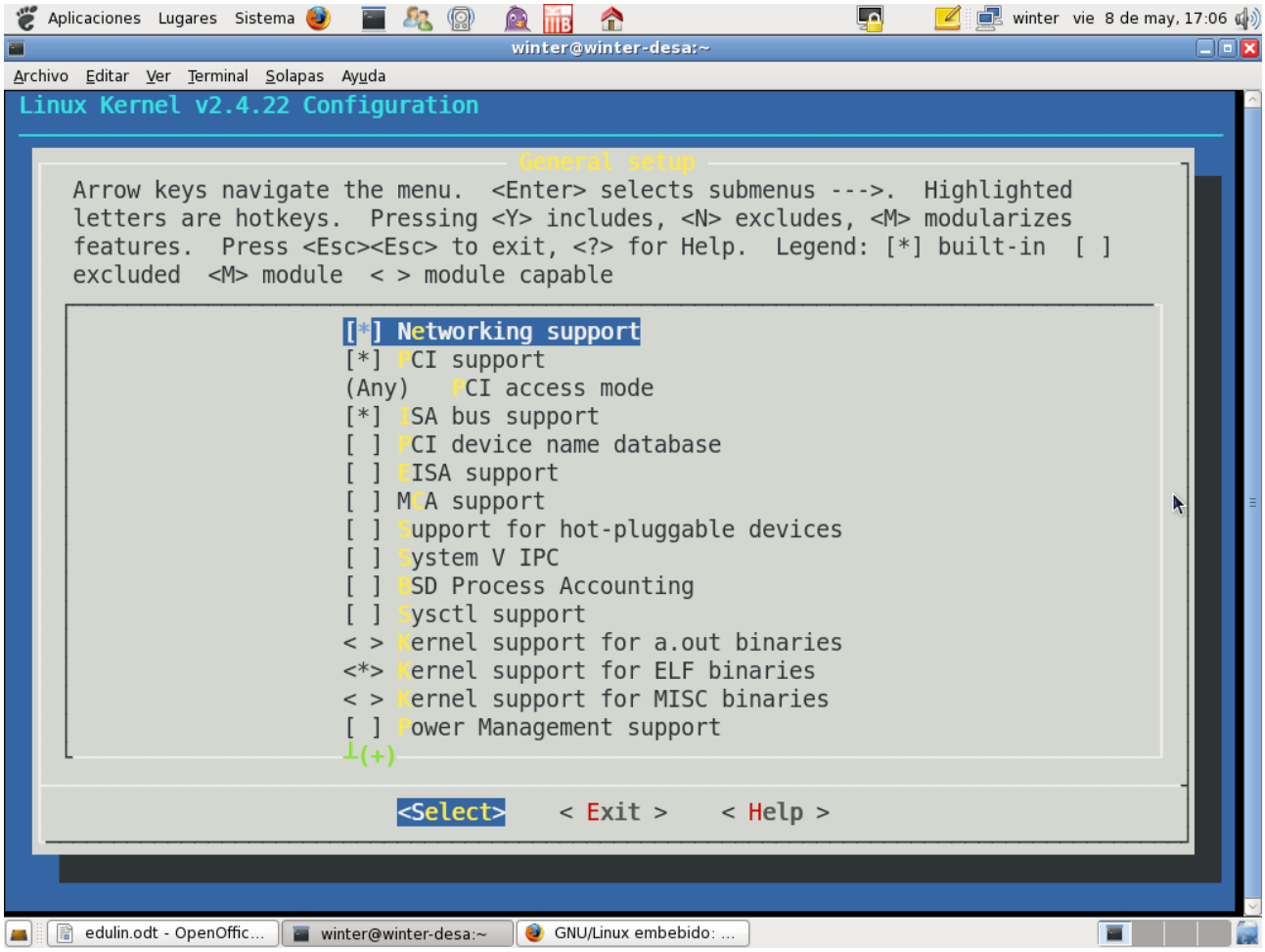
CONFIG_NOHIGHMEM - Según la documentación oficial: "*Linux puede usar 64GB de memoria en sistemas x86, sin embrago, el Bus de Direcciones de un microprocesador x86 de 32bit solo puede direccionar hasta 4GB... Si ud. nunca va a usar mas de 960MB de memoria, debería poner a off esta marca" (mas información en <Help>)*".

CONFIG_MATH_EMULATION - Según la documentación oficial: "*Linux puede emular un co-procesador matemático (usado para operaciones de punto o coma flotante) si el microprocesador destino no posee uno. Los 486DX y Pentium poseen en su infraestructura interna un co-procesador, pero los 486SX y 386 no, a no ser que se incluyan sus respectivos chips co-procesadores 487DX y 387.*"

Aquí le decimos que vamos a usar específicamente el microprocesador Intel 486, eliminando el soporte para memoria mas allá de 1GB (ya que nosotros solo vamos a usar 4MB) y nos cercioraremos de tener a mano un co-procesador matemático si llegamos a usar un micro Intel de la serie SX, el cual carecía de esta funcionalidad.

'GENERAL SETUP'

[*] Networking support
[*] PCI support
(Any) PCI access mode
[*] ISA bus support
<*> Kernel support for ELF binaries



CONFIG_NET - Según la documentación oficial: "*A NO SER QUE SEPAS MUY BIEN LO QUE VAS A HACER, deberías decir SI aquí La razón es que algunos programas necesitan que el Kernel tenga soporte para redes, para su correcto funcionamiento, aunque la maquina este funcionando físicamente aislada de una red*".

CONFIG_PCI - Según la documentación oficial: "*El Kernel busca si existe algún Bus PCI en la placa madre destino. PCI es el nombre de un tipo de Bus de sistema. Otros Bus de sistemas son: ISA, EISA, MICROCHANNEL (MCA) o VESA*".

CONFIG_PCI_GOBIOS - Según la documentación oficial: "*En sistemas PCI, la BIOS puede ser usada para detectar dispositivos que responden a estas características y determinar su*

configuración Lamentablemente algunas viejas placas madre PCI tienen errores de código en sus propias BIOS (buggy BIOS) y pueden colgar el sistema entero si son autodetectadas. Encima también puede suceder que algunas placas madres especificas para sistemas embebidos no poseen ningún tipo de BIOS. Linux, entonces, puede tratar de detectar los dispositivos PCI directamente sin el uso de la BIOS”.

CONFIG_BIOS - Según la documentación oficial: “*Linux tiene la capacidad de detectar si existen Slots ISA en su placa madre*”.

CONFIG_BINFMT_ELF – Se explica mas abajo.

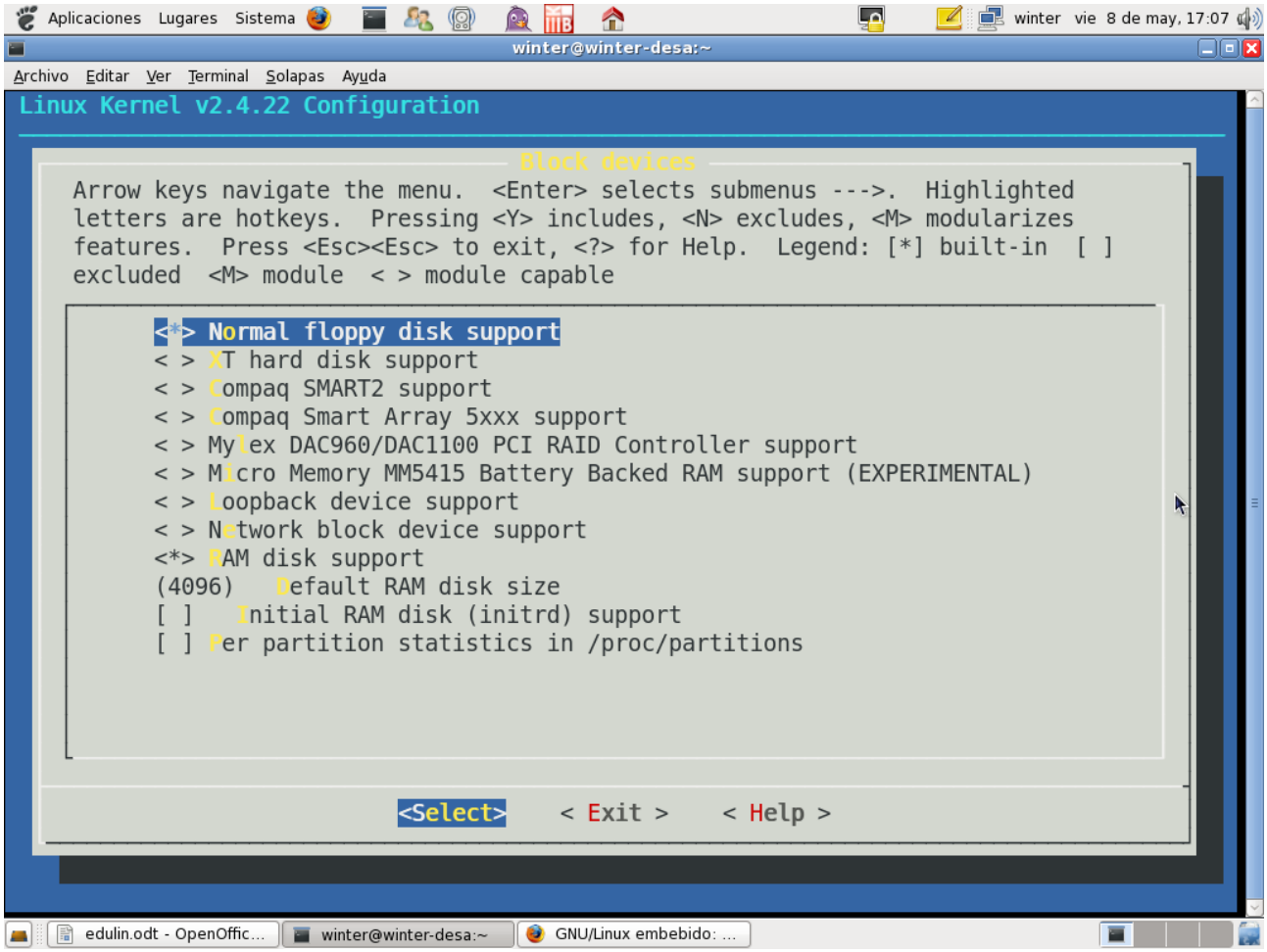
Obviamente, debemos habilitar el soporte de red, habilitar la lectura de puertos PCI, ya que las placas de video de este tipo de maquinas 486 solían ser onboard y traían su propio bus PCI. Con la opción 'any' le decimos al Kernel que si la BIOS falla al detectar el dispositivo PCI, le de lugar al núcleo para que el intente la autodetección. También incluimos soporte ISA, ya que la placa de red que vamos a usar tiene esa tecnología Por ultimo le permitimos al kernel ejecutar librerias en formato nativo ELF (Executable and Linkable Format).

'BLOCK DEVICES'

<*> Normal floppy disk support

<*> RAM disk support

(4096) Default RAM disk size



CONFIG_BLK_DEV_FD - Segun la documentacion oficial: *"Si quieres usar la diskettera, debes indicar SI aqui"*.

CONFIG_BLK_DEV_RAM - Segun la documentacion oficial: *"Diciendo SI aqui, permitimos al kernel usar una porcion de la memoria fisica como un dispositivo de bloque, con lo cual es permitido crear sistemas de archivos, leer o escribir en el mismo y cualquier otra cosa que normalmente se hace en un dispositivo en bloque (como un disco duro). Esta caracteristica es usada para albergar un root file system minimo."*

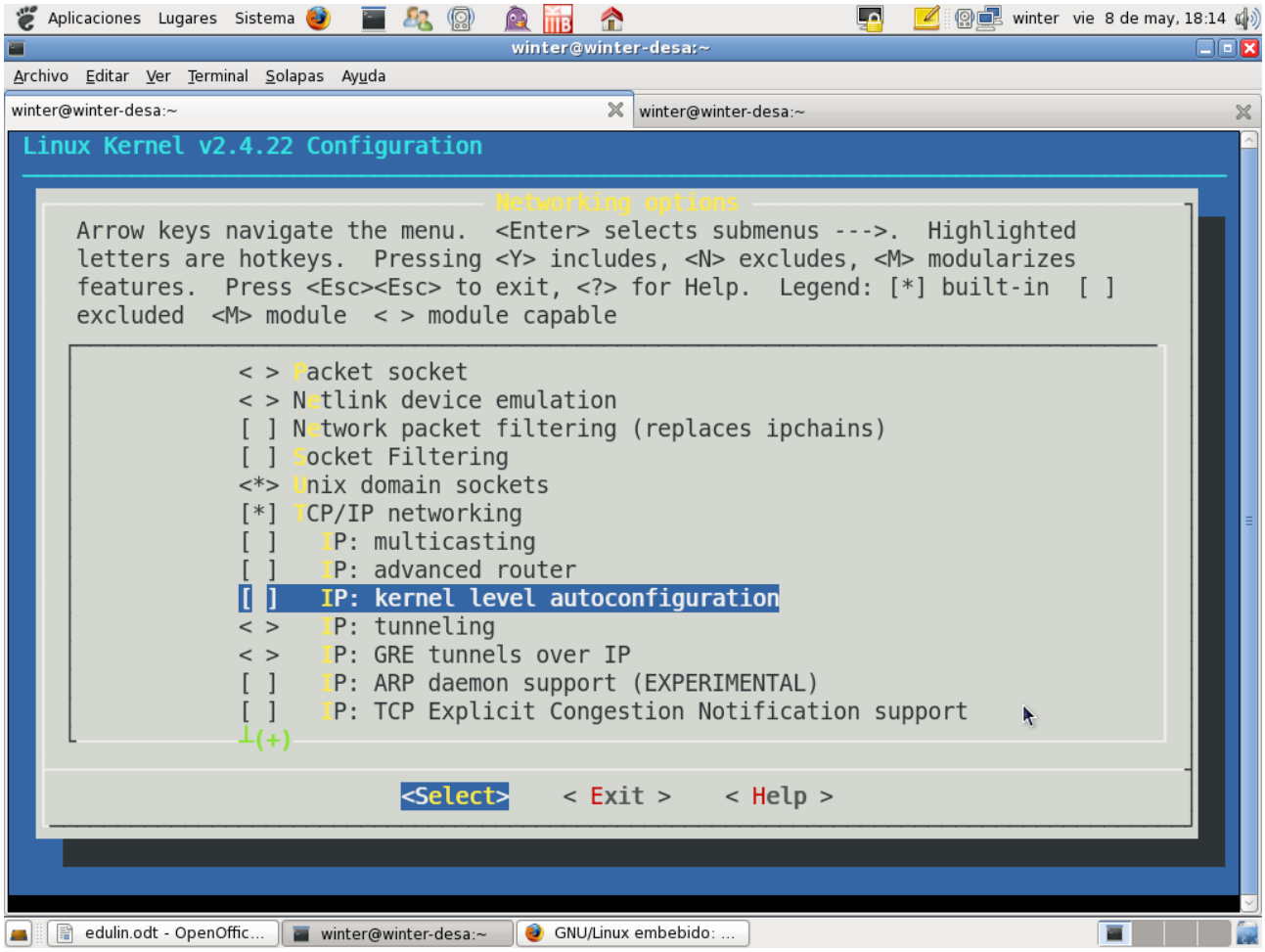
CONFIG_BLK_DEV_RAM_SIZE - Segun la documentacion oficial: *"El valor por defecto en 4096"*.

****NOTA: Aqui habilitamos la diskettera, que la vamos a usar al momento de cargar nuestro RFS. Ademas necesitamos especificarle al kernel donde colocar nuestro RFS, en memoria RAM, con una apertura minima de 4MB, correspondiente en este caso, a la cantidad de memoria fisica disponible en nuestro sistema".*

'NETWORKING OPTIONS'

<*> Unix domain sockets

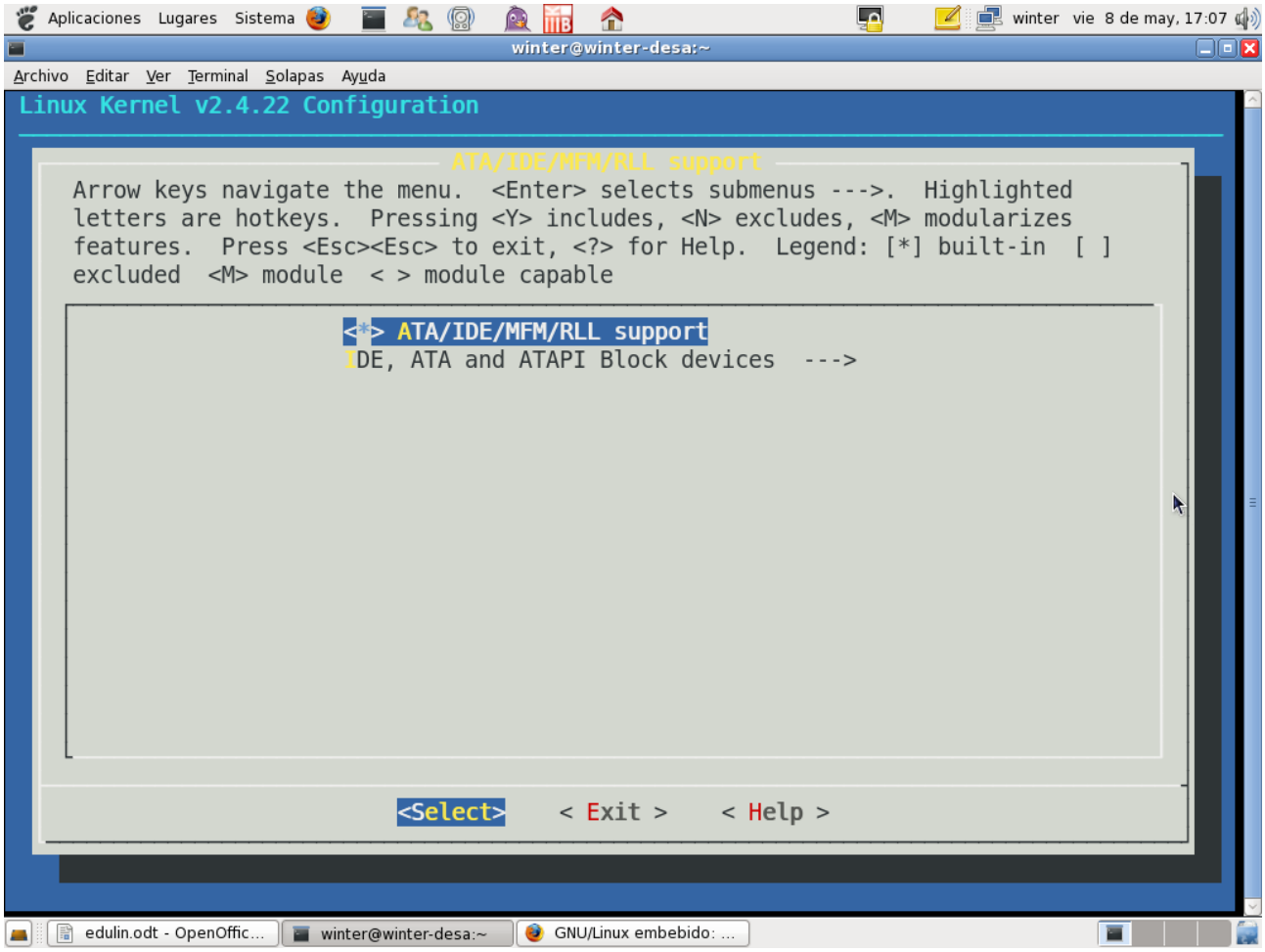
[*] TCP/IP networking



(***)CONFIG_UNIX - Segun la documentacion oficial: *"Diciendo SI aqui, ud incluye soporte de sockets para dominios UNIX. El SOCKET es el mecanismo standard de UNIX para establecer, acceder y conectarse a redes. Muy comunmente usado por programas como el sistema 'X Window' y 'syslog', los cuales usan los sockets de UNIX aun si la maquina no esta conectada a ninguna red".*

CONFIG_INET - Segun la documentacion oficial: *"Aqui estan albergados todo los protocolos usados en INTERNET y en la mayoria de las redes ETHERNET".*

'ATA/IDE/MFM/RLL SUPPORT'



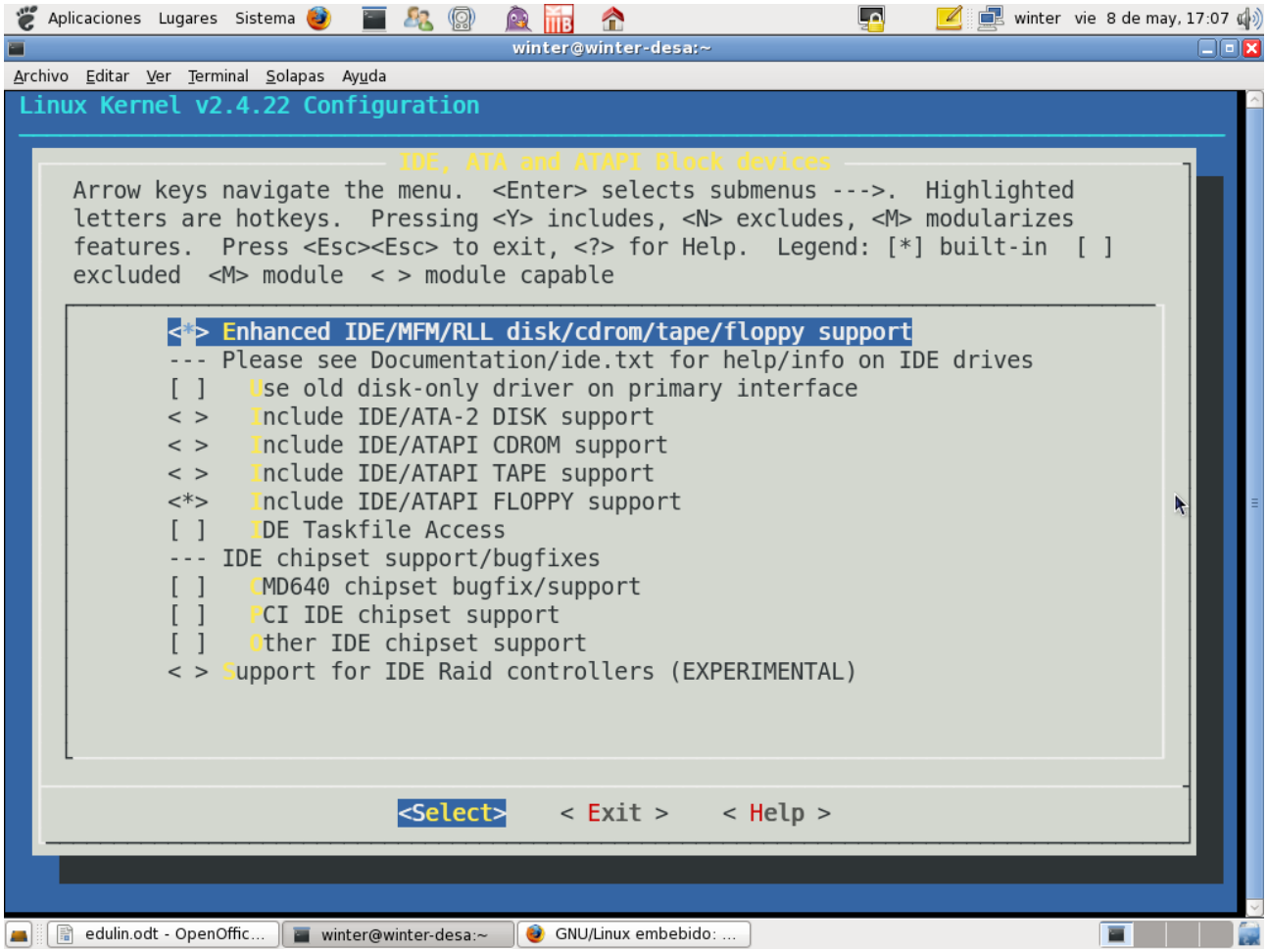
CONFIG_IDE - Segun la documentacion oficial: *"Con esta caracteristica habilitamos al kernel que pueda manejar unidades de almacenamiento tales como ATA/(E)IDE y ATAPI. Los casos mas comunes son los discos duros IDE y las unidades de CD-ROM ATAPIs. Integrated Disk Electronics (IDE alias ATA-1) es una especificacion de conectividad standard para unidades de almacenamiento masivo tales como discos duros. Fue diseñada por Western Digital and Compaq Computer en 1984. Tambien es llamada ST506.*
ATAttachment (ATA) es el superset de las especificaciones IDE.
ST506 fue tambien llamada ATA-1.

Fast-IDE es ATA-2 (tambien llamada Fast ATA), Enhanced IDE (EIDE) es ATA-3. Estas ultimas proveen soporte para discos grandes (hasta 8.4GB asi formando parte del LBA standard), mas discos (4 en vez de 2) y otras unidades de almacenamiento masivo tales como cintas magneticas y cdrom. UDMA/33 (aka UltraDMA/33) es ATA-4 y provee mas rapido (y mas amigable con el microprocesador) modos de transferencia que los anteriores PIO (Programmed processor Input/Output) de los anteriores standares ATA/IDE pertenecientes a los controladores DMA.

ATA Packet Interface (ATAPI) es un protocolo usado por las cintas y cd-roms EIDE, parecido en muchos aspectos al protocolo SCSI.
SMART IDE (Self Monitoring, Analysis and Reporting Technology) fue diseñado para prevenir la corrupcion de datos y rupturas de discos, detectando condiciones de pre-ruptura por hardware (calentamiento, tiempo de acceso, etc). Todos los discos construidos a partir de 1995 han debido

seguir este standard. El kernel en si no maneja esto, pero hay una gran cantidad de programas a nivel usuario tales como 'smart' que indaga en los parametros del standard SMART dentro del disco".

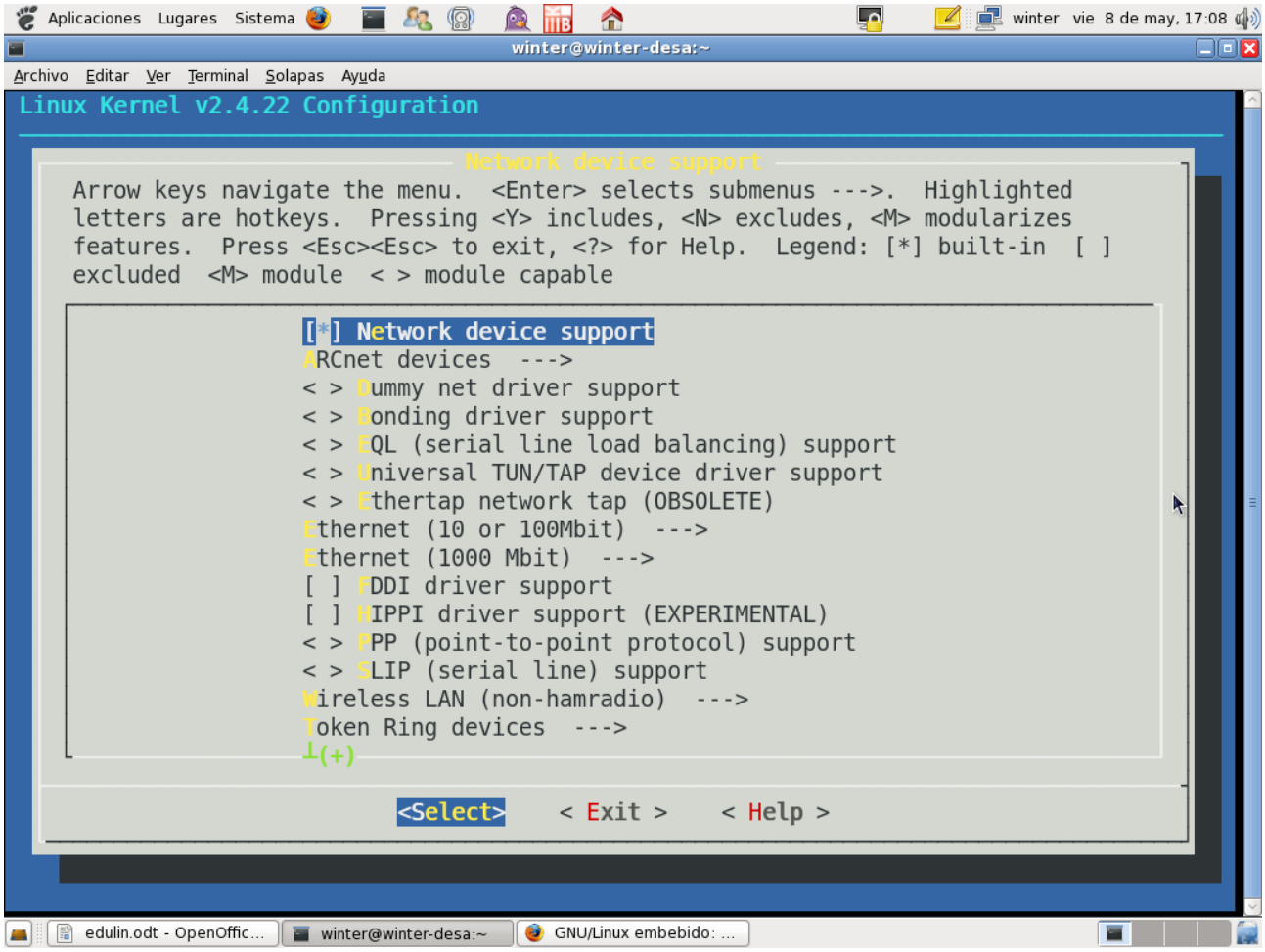
<*> ATA/IDE/MFM/RLL support
IDE, ATA and ATAPI Block devices --->
<*> Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support
<*> Include IDE/ATAPI FLOPPY support



CONFIG_BLK_DEV_IDE - Segun la documentacion oficial: *" Si dices que SI aqui, podras usar el total del conjunto de herramientas y caracteristicas para el manejo de sistemas de almacenamiento ATA/IDE y hasta un total de 10 interfaces 'MASTER' o 'SLAVE', cada una de ellas configurada como dispositivo o un total de hasta 20 ATA/IDE cdrom/tape/floppy".*

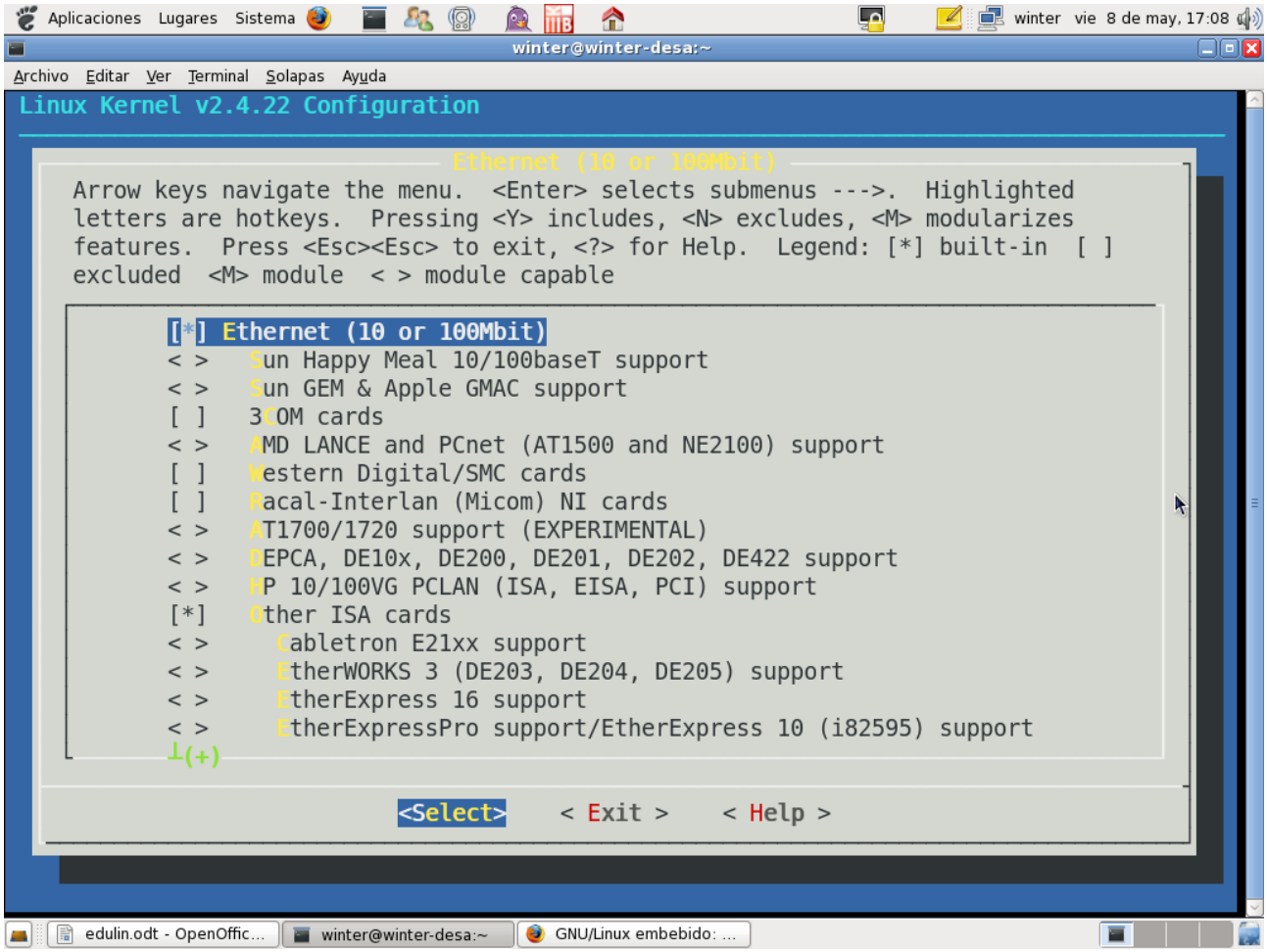
(***)CONFIG_BLK_DEV_IDEFLOPPY - Segun la documentacion oficial: *" Si tienes una diskettera la cual usa el protocolo ATAPI, responde que SI aqui".*

'NETWORK DEVICE SUPPORT'



CONFIG_NETDEVICES - Segun la documentacion oficial: *“ Puedes decir aqui NO si no deseas conectar tu computadora Linux con ningun otro dispositivo o si todas tus conexiones son realizadas por telefono a traves de un modem UUCP (UUCP es un protocolo para enviar mail y noticias entre hosts UNIX a traves de las lineas telefonicas, tambien usadas por las antiquisimas BBS de los años 70 y 80)”.*

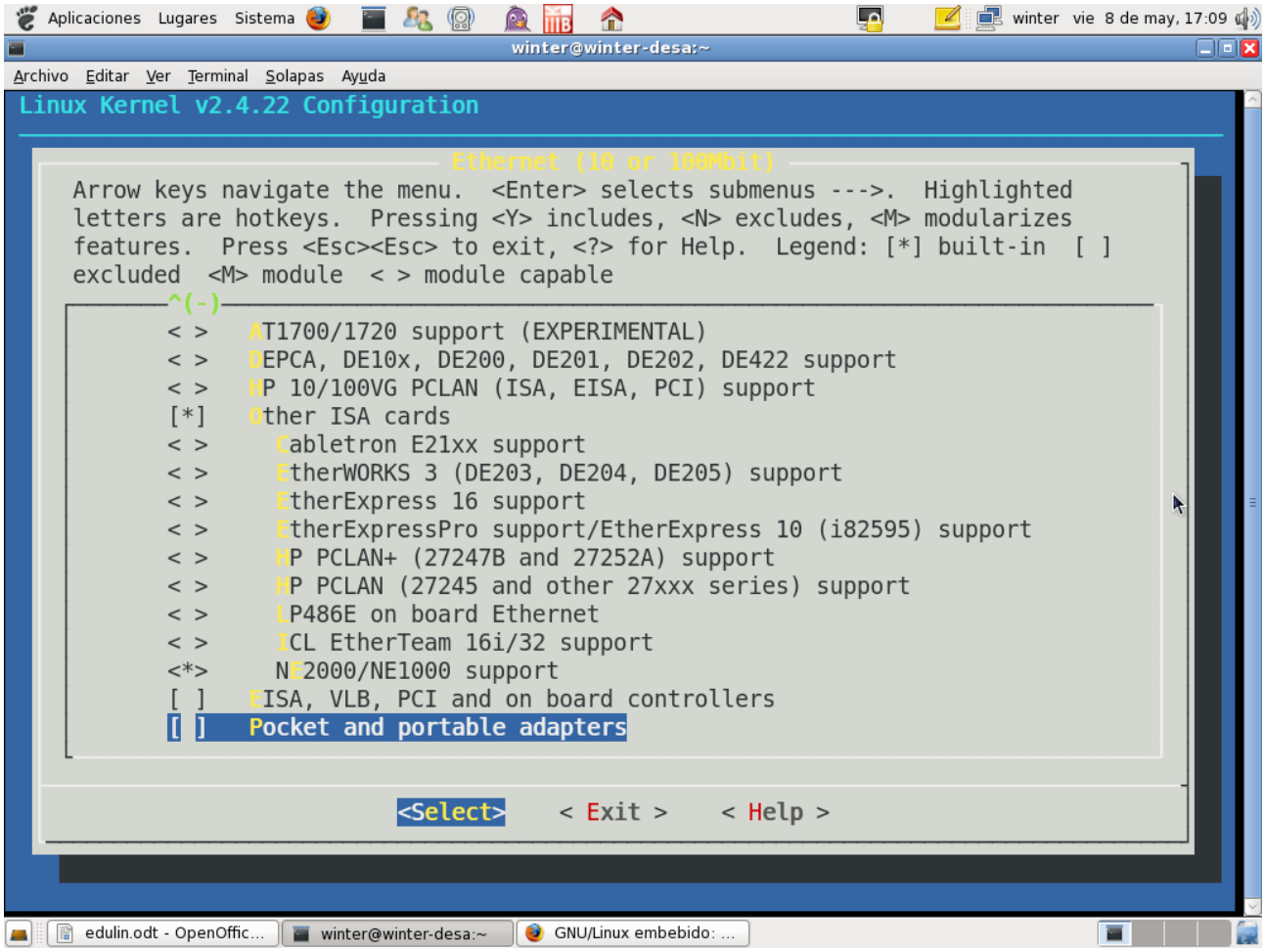
[*] Network device support
Ethernet (10 or 100Mbit) --->
[*] Ethernet (10 or 100Mbit)
[*] Other ISA cards



CONFIG_NET_ETHERNET - Segun la documentacion oficial: *“ Ethernet (tambien llamada IEEE 802.3 or ISO 8802-2) es el mas comun de los tipos de LANs (Local Area Network) dentro de las universidades y compañias.*
Las variedades comunes de Ethernet son: 10BASE-2 o Thinnet (10 Mbps en cable coaxil, computadoras linkeades en modo anillo), 10BASE-T o twisted pair (10 Mbps en cable UTP Categoria 5, computadoras linkeadas a un hub central), 10BASE-F (10 Mbps en fibra optica, computadoras linkeadas usando hubs), 100BASE-TX (100 Mbps en cable UTP Categoria 5, computadoras linkeadas a un hub central, 100BASE-T4 (100 Mbps sobre 4 pares trenzados standard para transmision de voz y datos, usando hubs). Las variedades de 100BASE son tambien conocidas como 'Fast Ethernet' y Gigabit Ethernet (1 Gbps sobre fibra optica o cobre)”.

CONFIG_NET_ISA - Segun la documentacion oficial: *“ Si su placa de red es ISA, por favor diga aqui que SI”.*

[*] Network device support
Ethernet (10 or 100Mbit) --->
[*] Ethernet (10 or 100Mbit)
[*] Other ISA cards
<*> NE2000/NE1000 support

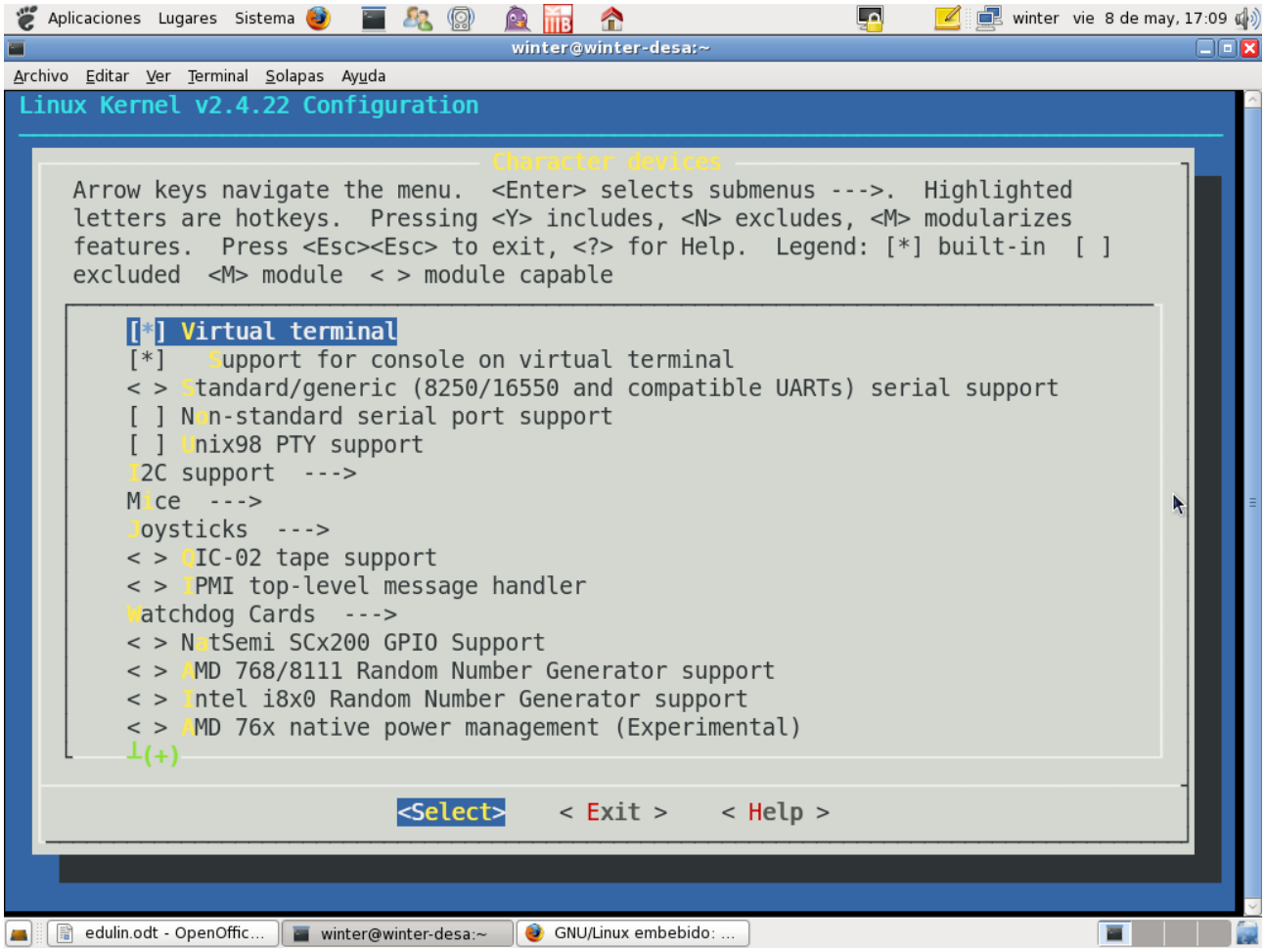


CONFIG_NE2000 - Segun la documentacion oficial: *“ Si ud. tiene una placa de red del tipo NE2000, diga aqui que SI. Muchisimas placas con chipsets propietarios o especificos son totalmente compatibles con NE2000”.*

'CHARACTER DEVICES'

[*] Virtual terminal

[*] Support for console on virtual terminal



CONFIG_VT - Segun la documentacion oficial: *“ Acceso a dispositivos que soportan terminales con elementos adosados tales como un display y un teclado. Estas tambien son llamadas 'virtuales' porque soportan multiples terminales virtuales o consolas en una unica terminal fisica (en este caso nuestra pantalla y teclado). Esto es muy util a la hora, por ejemplo, de recolectar mensajes de sistema y alertas en una consola virtual, mientras que en otra podemos tener acceso en modo usuario – modo texto y una tercera corriendo un servidor X, todas en paralelo. Cambiar entre consolas es posible utilizando las teclas de funcion mas la tecla 'Alt' (ejemplo: Alt+F1)”.*

CONFIG_VT_CONSOLE - Segun la documentacion oficial: *“ el sistema de consola es el dispositivo que recibe todos los mensajes del kernel y advertencias, permitiendo ingresar a ella en modo monousuario”.*

'CHARACTER DEVICES'

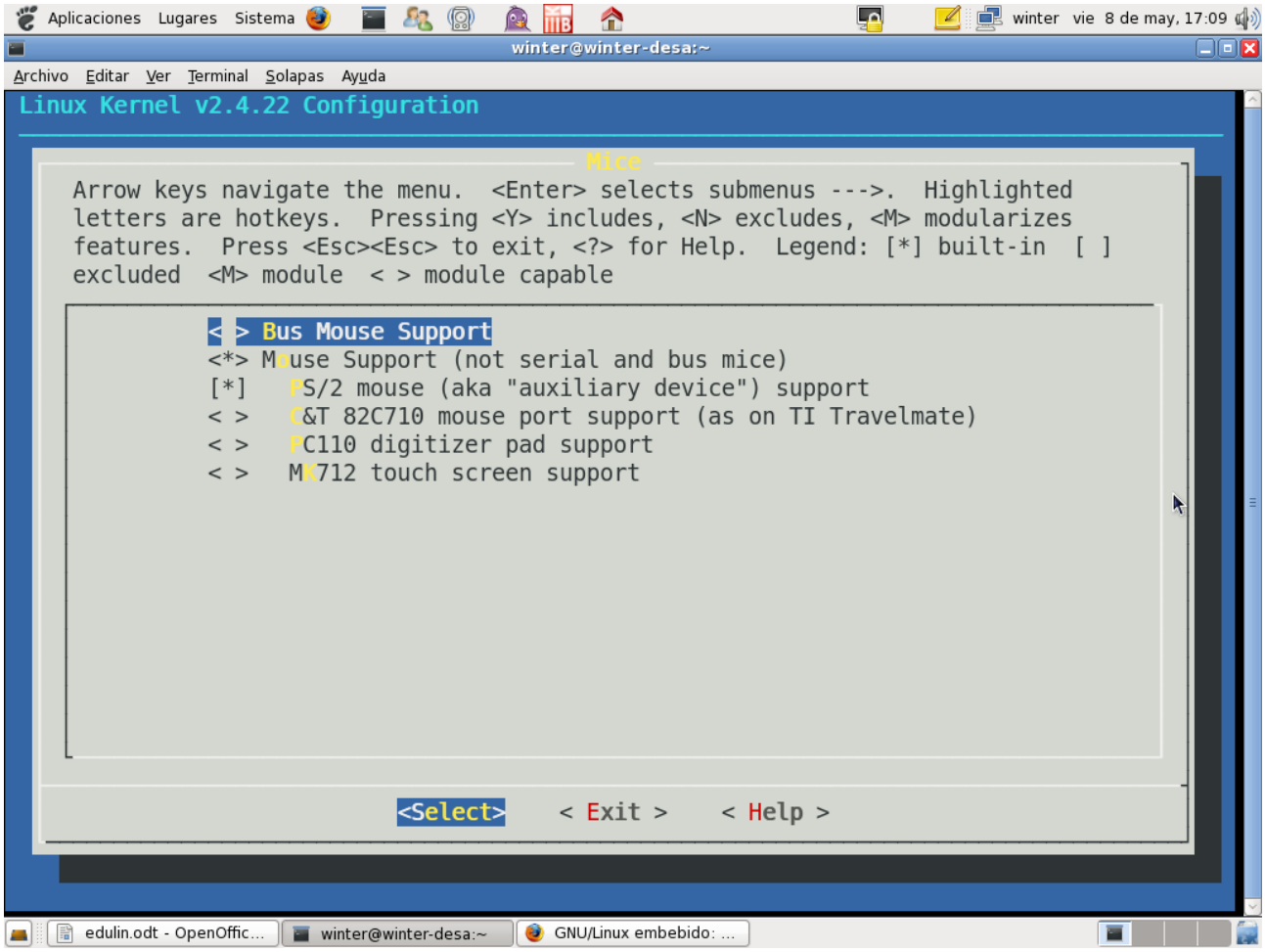
[*] Virtual terminal

[*] Support for console on virtual terminal

Mice --->

<*> Mouse Support (not serial and bus mice)

[*] PS/2 mouse (aka "auxiliary device") support

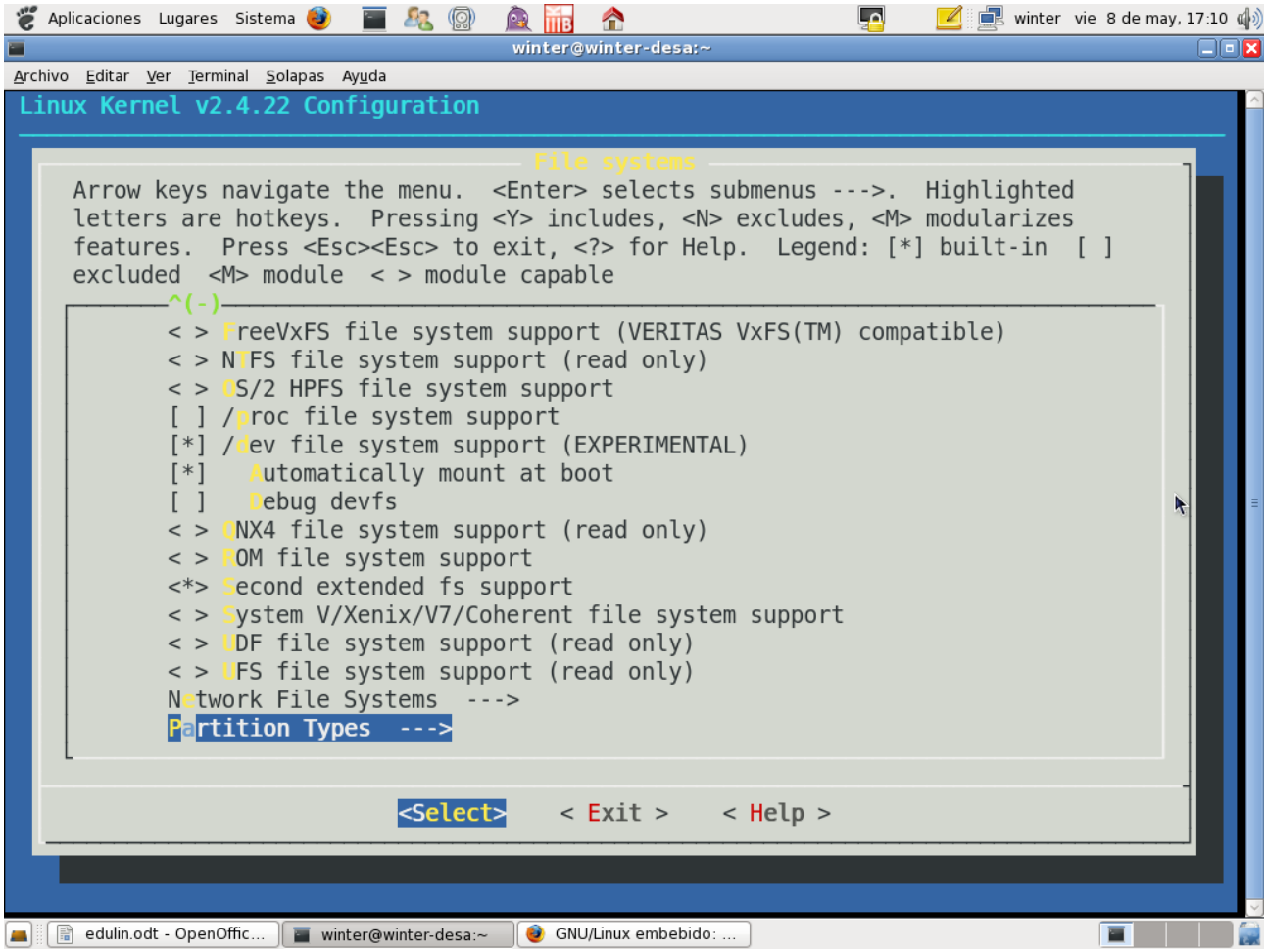


CONFIG_MOUSE – segun la documentacion oficial: *"Esto es para maquinas con mouses que no son ni seriales ni bus. Ejemplos de esto son los mouse PS/2 y algunas tabletas digitalizadoras. La mayoria de las personas tienen un mouse serial regular , MouseSystem o mouse Microsoft (construido por Logitech), que se enchufa dentro de un puerto COM DB-9 o DB-25".*

CONFIG_PSMOUSE: segun la documentacion oficial: *"Los mouse PS/2 conectan a un puerto especial que se parece mucho mas a un puerto de teclado (pequeño conector circular de 6 pines o MINI DYN). De esta forma no se utuliza el puerto serial standard. Este puerto tambien puede ser usado para otros dispositivos tales como lapiz optico, tabletas, etc. COMPAQ, EPSON, AT&T e IBM usan estos puertos para la manufactura de sus maquinas".*

'FILE SYSTEMS'

[*] /dev file system support (EXPERIMENTAL)
[*] Automatically mount at boot
<*> Second extended fs support



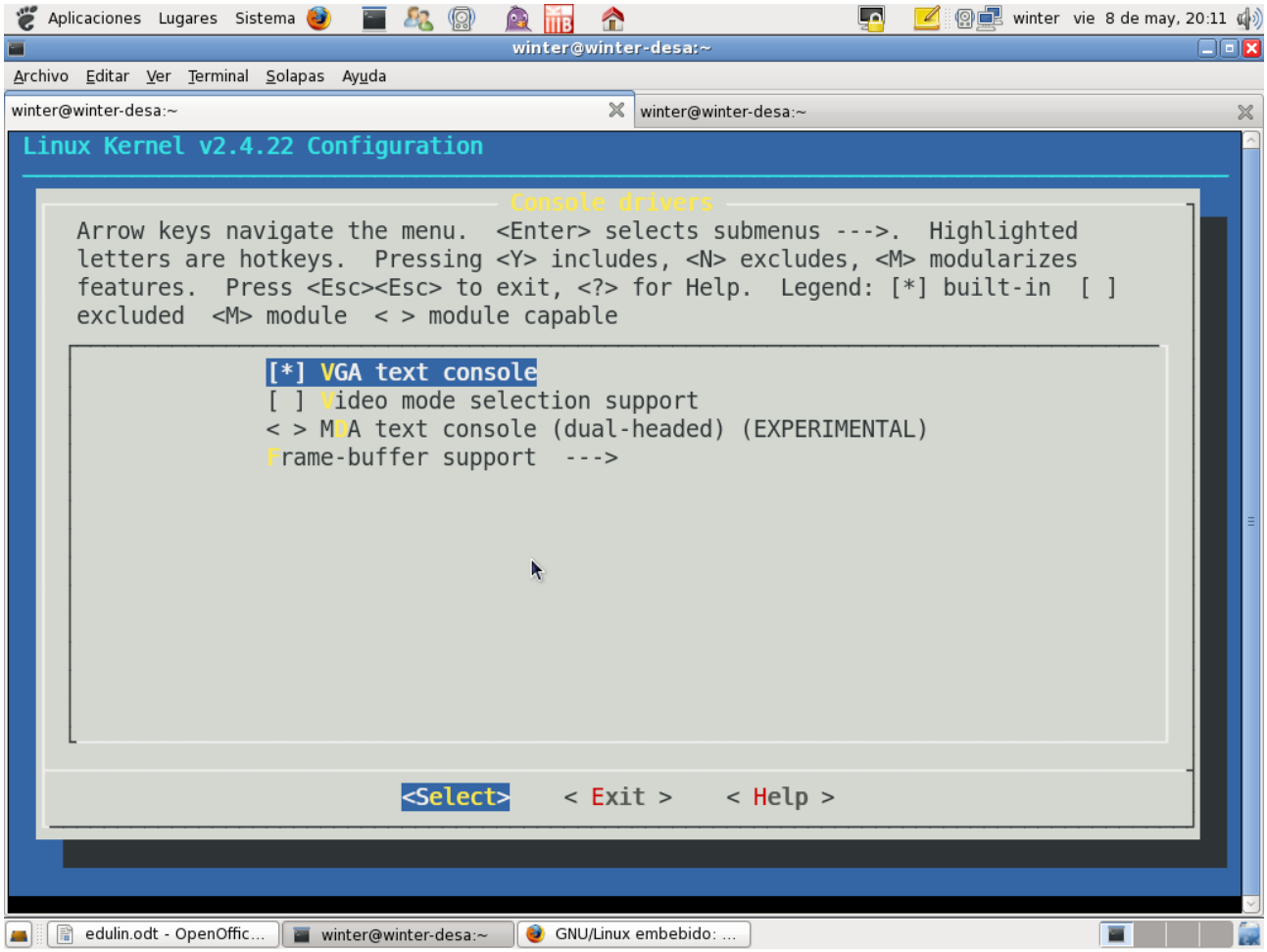
CONFIG_DEVFS_FS - Segun la documentacion oficial: *“ Este es el soporte para DEVFS, un sistema de archivos virtuales, al igual que /proc el cual provee la interface de sistema de archivo para distintos dispositivos, encontrados normalmente en /dev. Devfs no depende de locaciones de numeros mayores o menores. Los dispositivos que registren entradas en /dev, son mostrados inmediatamente, eso significa que el administrador del sistema no tiene que crear puntos de montaje estaticos para cada posible dispositivo (block, caracter, especial) usando el comando mknod o el script MAKEDEV nunca mas”.*

CONFIG_DEVFS_MOUNT - Segun la documentacion oficial: *“ Esta opcion solo aparece si ONFIG_DEVFS_FS esta habilitado. Diciendo SI aqui, haremos que el kernel automaticamente monte defs dentro de /dev cuando el sistema arranca, antes que el hilo de init comience”.*
CONFIG_EXT2_FS - Segun la documentacion oficial: *“ Este de hecho es el sistema de archivos nativo de Linux para disketteras y discos duros”.*

****NOTA: Por defecto a cualquier compilacion standard de kernel se le incluye el pseudo sistema de archivos '/proc', el cual es una extension al usuario de las actividades del kernel en forma de archivos de texto escritos al vuelo, con especificacion detallada del sistema.* *En este caso particular /proc consume preciosa memoria que no disponemos, asi que lo he*

desabilitado.
*Esta desicion no esta publicada anteriormente en ningun documento creeria yo. De todas formas la falta de /proc no es critica para el Kernel.****

'CONSOLE DRIVERS'
[*] VGA text console



CONFIG_VGA_CONSOLE - segun la documentacion oficial: *"Diciendo SI aqui, permitimos a Linux que use una consola en modo texto, utilizando las librerias VGA standard"*.

Finalizando el nuevo KERNEL

Es hora de compilar nuestro nuevo kernel, para eso hacemos lo siguiente:

```
$ make dep
$ make bzImage
```

Con **dep** preparamos las cabeceras de linux y todas sus dependencias, con **bzImage** creamos la imagen en si. Tiempo estimado de compilacion 5 a 7 minutos en un pentium 3 con 256MB RAM.

He decidido construir un kernel monolitico en su totalidad para evitar el uso de modulos sueltos, ya que de esta forma minimizo el error de portabilidad del mismo.
De esta forma, conseguimos una bzImage de =~509KB.

La imagen resultante se encuentra en arch/i386/boot, ingresar a este directorio:

```
$ cd arch/i386/boot
```

Copiar la imagen a nuestro directorio de trabajo:

```
$ cp bzImage ~
```

Construyendo uClibC

Es hora de compilar uClibC...
Nos posicionamos en el directorio de uClibc:

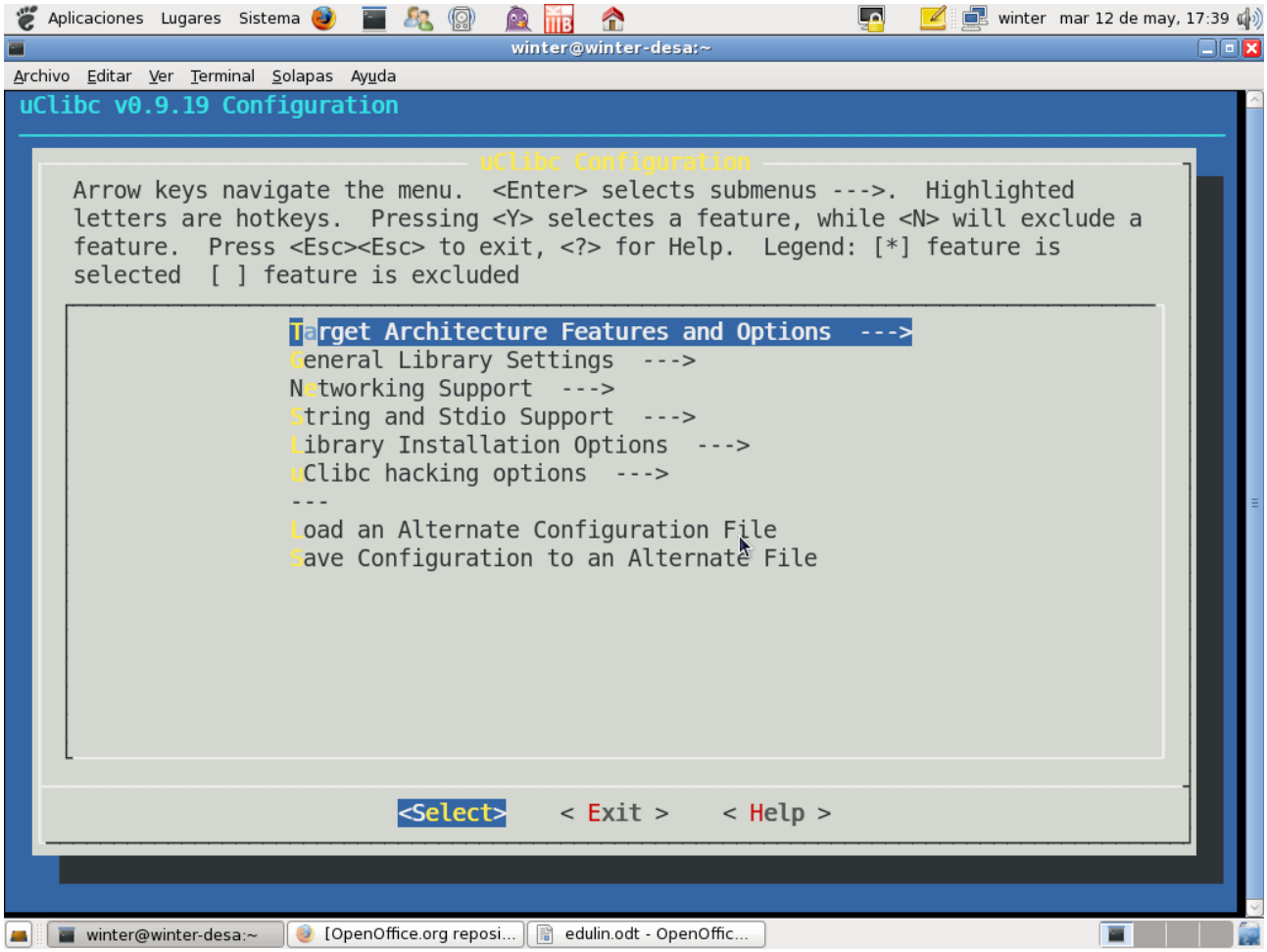
```
$ cd ~/uClibc-0.9.19
```

Anteriormente con el kernel, nosotros optimizamos en tamaño la resultante compilacion, en el caso de uClibC, este trabajo se realiza de forma automatica.
Quiero destacar que a esta libreria la vamos a instalar en el sistema anfitrion, junto con las demas librerias del sistema, para esto debemos trabajar como superusuarios, pero con las variables de entorno del usuario 'bootdisk':

```
$ su
#
```

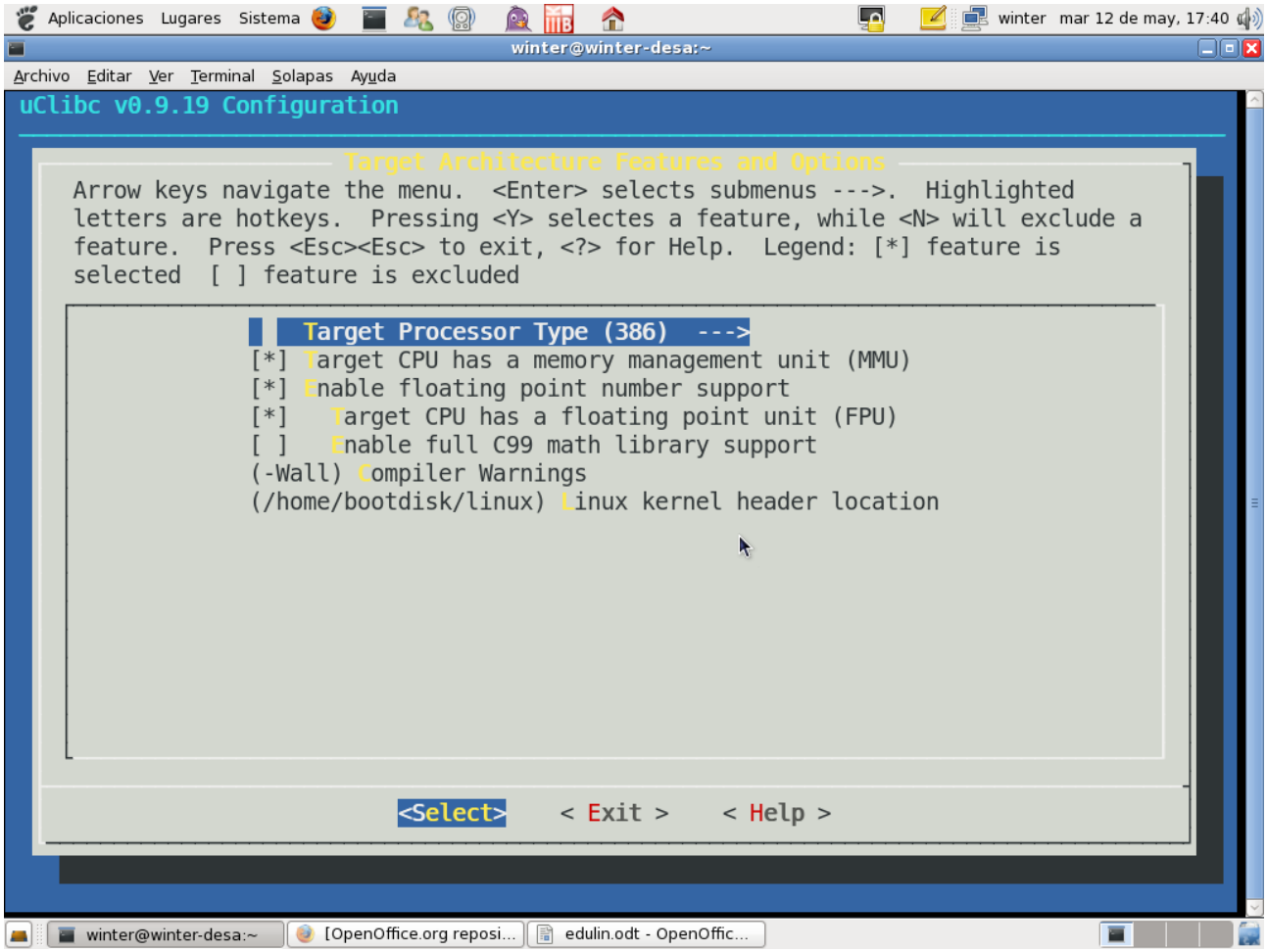
ejecutamos la herramienta de configuracion grafica que trae por defecto similar a la del kernel:

```
# make menuconfig
```



Como podemos apreciar, esta es la pantalla del menu general de opciones, simplemente configuramos tres variables de entorno necesarias para poder compilar de forma correcta nuestra uClibC:

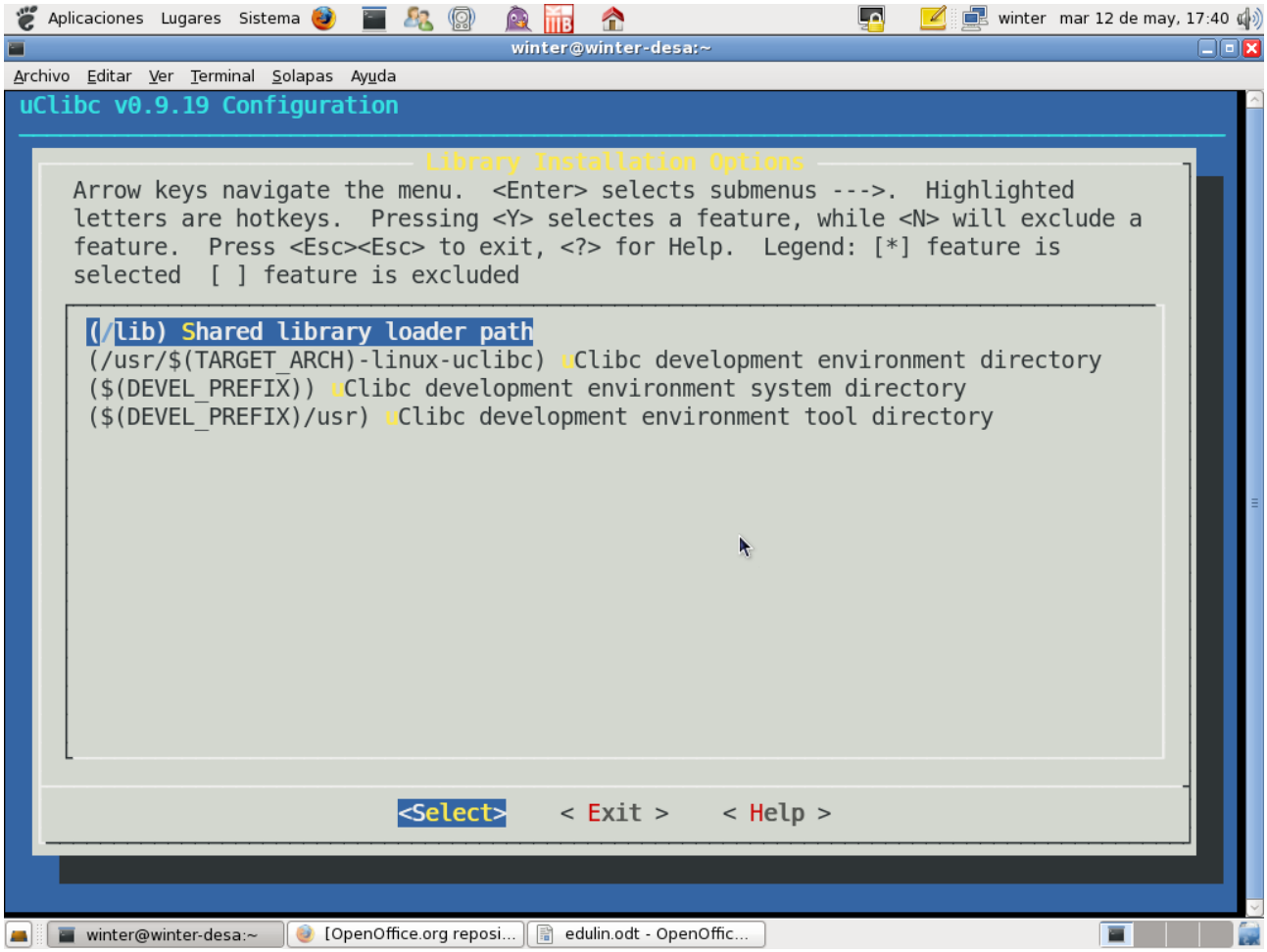
- Target processor type
- Linux kernel header location
- Shared library loader path




```
Target Architecture Features and Options  --->
                    Target Processor Type (386)  --->
                          Linux kernel header location --->
                                                  (/home/bootdisk/linux)
```

Indicamos el tipo de microprocesador que vamos a usar, en este caso arquitectura 386 y el path donde se encuentra en este momento las fuentes de nuestro kernel recien compilado.

***NOTA: ¡CUIDADO! TANTO EL TIPO DE PROCESADOR ESPECIFICADO A LA HORA DE COMPILAR EL KERNEL, COMO EL TIPO DE PROCESADOR ESPECIFICADO A LA HORA DE COMPILAR UCLIBC DEBEN SER LOS MISMOS, o sea que si yo compile el kernel con arquitectura 386, en la configuracion de uClibC debe corresponderse tambien este mismo microprocesador, o sea, 386.
El resultado de no 'matchear' las dos configuraciones nos impedira compilar correctamente la libreria y ademas nos sera imposible compilar correctamente svgalib.***




```
Library Installation Options  --->
          Shared library loader path --->
                                        (/lib)
```

Indicamos donde se encuentran en nuestro sistema el conjunto de librerias compartidas.

Una vez que terminamos guardamos los cambios y salimos.
El lugar donde de ahora en mas se encontrara instalada esta libreria es en /usr/i386-linux-uclibc.

Compilar e instalar:

```
# make
# make install
# exit
```

Creando nuestro Root file system

El RFS es el esqueleto de cualquier sistema tipo unix/linux, sin el es imposible hablar siquera que es un sistema operativo, vamos a utilizar la jerarquia standard de FHS como inspiracion para nuestro RFS. Vamos a crear un directorio contenedor, un pseudo '/' en donde hacer un chroot mas adelante para comprobar que esten funcionando correctamente las aplicaciones base de nuestro RFS llamado 'rfs':

```
$ cd
$ mkdir rfs
$ mkdir -p rfs/{dev,proc,etc/init.d,sbin,bin,lib,mnt,usr,var/log}
$ cat > rfs/etc/init.d/rcS << "EOF"
#!/bin/sh
EOF
$
```

El archivo etc/init.d/rcS es el proceso 'init' encargado de ejecutar el inicio de los servicios del sistema operativo, por el momento esta vacio, pero pronto incluiremos comandos para ejecutar las respectivas tareas.
Ahora debemos hacer ejecutable este archivo, sino, al no ser ejecutable, el kernel no lo interpreta como tal y no hay posibilidades de arrancar el sistema en si:

```
$ chmod 755 rcS
```

Hemos creado un esqueleto de directorios minimo para que cumpla estrictamente la funcion de albergar solo lo necesario.

****NOTA: aunque aqui declaramos el directorio 'proc' y aunque originalmente en el documento de Csaba Henk, este se usaba para almacenar informacion del kernel (uso de memoria, dispositivos activos, etc), no lo utilizamos en esta nueva implementacion, ya que consume una importante cantidad de memoria escribiendo al vuelo datos e informacion referente a procesos de sistema. Por lo tanto, el soporte de archivos /proc fue eliminado de la configuracion del kernel. Ademas no es critica su ausencia, el kernel igual puede funcionar exactamente como lo hace con el soporte a /proc, sin ningun problema, solo nos quedamos ciegos al no poseer ninguna informacion de la situacion actual del mismo (accesos a memoria, consumo, etc). Actualmente lo dejo por compatibilidad con FHS.****

Instalando uClibC en nuestro Root file system

Es momento de copiar el resultado de la construccin de uclibc a nuestro nuevo entorno:

```
$ cp -a /usr/i386-linux-uclibc/lib/{ld-
uClibc*,libc.so.0,libuClibc-*} \
 ~/rfs/lib
```

****NOTA: si no funciona el copiado, o sea, obtenemos errores de permisos, copiar con 'su'.****

Construyendo BUSYBOX

Es hora de compilar y construir nuestra navaja suiza todoterreno, la cual contendra las aplicaciones necesarias para poder inicializar de forma correcta nuestro sistema operativo.

Nos posicionamos dentro del directorio de las fuentes:

```
$ cd ~/busybox-0.60.5
```

hacemos un backup del archivo de configuracion que trae por defecto, por si las cosas no salen del todo bien y es necesario recompilar desde cero:

```
$ cp Config.h Config.h.old
```

Ahora es tiempo de configurar con nuestro editor favorito (para mi vim) este archivo de configuracion. Dejo las opciones que yo he habilitado en mi archivo de configuracion como posible guia.

****NOTA: Cada linea representa una caracteristica, las lineas estan comentadas al estilo C++ con '//', eliminando estas dos barras simples, habilitamos las opciones.Config.h:****

```
#define BB_ASH
#define BB_DATE
#define BB_ECHO
#define BB_IFCONFIG
#define BB_INIT
#define BB_INSMOD
#define BB_LS
#define BB_PING
#define BB_TTY
#define BB_UNAME
// End of Applications List
//
// If you enabled one or more of the shells, you may select which
one
// should be run when sh is invoked:
#define BB_FEATURE_SH_IS_ASH
// Make init use a simplified /etc/inittab file (recommended).
#define BB_FEATURE_USE_INITTAB
//Enable init being called as /linuxrc
```

```
#define BB_FEATURE_LINUXRC
// Support insmod/lsmod/rmmod for post 2.1 kernels
#define BB_FEATURE_NEW_MODULE_INTERFACE
// Enable ifconfig status reporting output -- this feature adds
7k.
#define BB_FEATURE_IFCONFIG_STATUS
// Support for devfs.
#define BB_FEATURE_DEVFS
//
// End of Features List
```

Con esto tenemos habilitadas las herramientas minimas para trabajar en caso de fallas de conexion de la red, saber la fecha y hora, version de kernel y por supuesto el interprete de comandos (ash en este caso).

Es hora de compilar e instalar...
Antes que nada, nos cercioraremos que la variable $HOME es la correcta, esta debe apuntar hacia nuestro directorio de trabajo que por defecto es /home/utc. Si este no es el caso escribimos:

```
$ export $HOME=/home/utc
```

Manos a la obra:

```
$ make CROSS=i386-uclibc-
$ make PREFIX=$HOME/rfs install
```

****NOTA: no olvidar el simbolo (-) al final de la variable CROSS****

La variable CROSS le indica a make que debe usar las librerias recien instaladas de uClibC que se encuentran como sabemos en /usr/i386-linux-uclibc.

La variable PREFIX le indica a make donde va a copiar los archivos creados por el mismo en el paso anterior.

Comprobando la buena salud de BUSYBOX

Es momento de parar y hacer nuestro primer test para comprobar si nuestras herramientas contenidas en busybox han sido correctamente compiladas de cara a la libreria uClibC:

```
$ cd ~
$ su
# chroot rfs /bin/sh
```

Debemos obtener la siguiente leyenda:

```
BusyBox v0.60.5 (2009.04.15-22:40+0000) Built-in shell (ash)
Enter 'help' for a list of built-in commands.

# ls
# exit
```

```
# exit
$
```

Como podemos observar, con chroot, le decimos al sistema que reemplacemos nuestro root file system (alojado en /) por un nuevo arbol dependiente de un nuevo root (ahora alojado en /home/bootdisk/rfs/) alojado en nuestro directorio contenedor 'rfs', le damos la orden que ejecute un nuevo interprete de comandos, en este caso es 'sh' el cual es un enlace blando al ejecutable busybox que se encuentra en /bin del nuevo chroot, el cual en realidad invoca a 'ash' un interprete de comandos minimalista nativo de busybox.

****NOTA: vemos que tenemos dos veces el mismo comando 'exit', esto es asi porque no hay que olvidar que estamos en un entorno chroot ajeno al nuestro por defecto, asi que primero salimos del entorno de busybox y luego damos de baja la cuenta de superusuario para volver a encontrarnos con nuestro usuario por defecto (en este caso 'utc').****

CAPITULO 3: Implementando el acceso remoto gráfico

De la caja a la pantalla

Hasta ahora lo que hemos logrado es tener una pequeña cajita con un kernel minimo y un set casi ausente de funcionalidades para nuestro microsistema, el cual de esta forma, realmente no nos sirve de mucho.
De todas formas, estos pasos anteriores te serviran para entender como podemos construir un sistema GNU/LINUX desde cero, que requiera de consola unicamente. He eliminado la mayoria de las funcionalidades de busybox por razones obvias, ya que lo unico que necesito es tener un entorno grafico inmediatemente despues de haber cargado la ultima linea de codigo de mi sistema de consola.

Por eso mismo, aqui comienza la segunda etapa de construccion.
Hemos comprobado que todo funciona correctamente con la ayuda de chroot, ahora debemos continuar con la compilacion del codigo restante.

Como dije anteriormente, el truco para obtener un entorno grafico atractivo y actual con solo 4MB de RAM es gracias al trabajo conjunto de SVGAlib y svncviewer, el cual en este caso nos entregaran una pantalla con una profundidad de colores de 15 bits (32768 colores) y una dispocision de 800*600 pixeles (aunque es perfectamente viable tener una profundidad de color de hasta 16bits, dado que la memoria de video de este tipo de hardware es de 1MB).

****NOTA: Una pregunta que me he hecho alguna vez y me han hecho es: '¿cual es la maxima resolucion para mi placa de video?', bien, en este caso te muestro la forma de saber cuanta memoria ocupa un tipo de resolucion con una profundidad de color determinada, que en este caso usaremos la tipica resolucion de 800 pixeles de ancho por 600 pixeles de alto con una profundidad de colores de 16bits. La formula es muy simple:*

```
(ResHoriz * ResVert * ProfCol) / 8 = Memoria necesaria en bytes

Esto lo podemos ver asi entonces:

(800 * 600 * 16) / 8 =960000 bytes

Si a este resultado lo convertimos en KiloBytes:

960000 / 1024 = 937.5 KB
```

*Ahora podemos ver que para una resolucion de 800*600*16, como minimo necesitamos 1MB de memoria de video.****

Instalando las librerias matematicas en rfs

Para poder usar el mouse, es necesario tener las librerias matematicas que se generan cuando compilamos uClibC, estan se encuentran en su directorio lib/, copiamos:

```
$ cd /usr/i386-linux-uclibc/lib
$ cp -a libm-* libm.so.0 ~/rfs/lib
```

****NOTA: si no nos permite copiar las librerias nuestro sistema, podemos usar la sentencia 'su' como siempre.****

Configurando e instalando SVGAlib

Antes que nada, necesitamos parchar las fuentes de svgalib, ya que en esta versión encontramos errores ortograficos, solucionados en las posteriores.
Asumiendo que nuestro patch ya esta en el directorio de svgalib parchamos todo, si todavia no, lo copiamos dentro de la carpeta:

```
$ cd ~
$ cp sources/svgalib-1.9.18-demos_Makefile-1.patch svgalib-1.9.18
$ cd svgalib-1.9.18
$ ln -s ../egadrv.c src/drivers
$ patch -Np1 -i svgalib-1.9.18-demos_Makefile-1.patch
```

Aqui copiamos el parche, luego antes de ejecutarlo, creamos un enlace blando al fuente egadrv.c que esta en el raiz de svgalib desde src/drivers, ya que es un enlace necesario y no ha sido declarado. Por ultimo parchamos.

Ajustando nuestro Makefile

Es momento de ajustar nuestro Makefile.cfg el cual alberga la informacion necesaria para sooportar distintas placas de video, aseguro que esta version soporta una importante cantidad.

Antes que nada, debemos revisar la placa del hardware destino.
La idea es elegir solo el dispositivo que vamos a utilizar, en este caso he utilizado a Cirrus Logic, ya que la mayoria de las placas madre de maquinas tales como IBM, AT&T, EPSON, HP y COMPAQ poseian este chip de video. En este caso, usando solo una placa de la cantidad que trae declaradas, he conseguido un binario de 129KB.

NOTA: ¿Como se cual es la placa de video del dispositivo cliente? Basicamente, destapamos la maquina y ubicamos el chipset de video, el cual es facilmente identificable por ser una cuadrado negro de aproximadamente 4cm por lado, encontrando las siguientes leyendas de ejemplo: Trident, Cirrus Logic, etc.

Configuracion personalizada de SVGAlib (Makefile.cfg)

Solo muestro lo que debemos descomentar:

```
#
# Comment out any driver that you don't want included in the
library.
#
INCLUDE_CIRRUS_DRIVER = y
#
# Comment out any adapter you don't want to autodetect.
#
INCLUDE_CIRRUS_DRIVER_TEST = y
```

Aqui le decimos que incluya el driver para las placas Cirrus Logic y que habilite todos los modos de

video disponibles para esta placa de video.
Tambien sugiero que se lean bien los enunciados de este Makefile, ya que estan muy entendibles.
Comentar la creacion de las 'man' tambien por ejemplo, asi inhibimos su creacion.

Ahora compilaremos e instalaremos las librerias resultantes dentro de usr/lib ubicado dentro de $HOME/rfs:

```
$ export KERNEL=/home/utc/linux
$ make shared
$ cd sharedlib
$ for q in “” gl; do
>strip libvga$q.so.1.9.*
>ln -s libvga$q.so.1.9.* libvga$q.so.1
>ln -s libvga$q.so.1 libvga$q.so
>done
$ mkdir -p $HOME/rfs/usr/lib
$ cp -a libvga* $HOME/rfs/usr/lib
```

Lo que hemos hecho aqui es declarar la variable $KERNEL con su path absoluto, para indicarle a make donde se encuentran las cabeceras del mismo, luego creamos las librerias compartidas.
La parte mas interesante de esto esta en el ciclo 'for' el cual busca todas las librerias que contengan en el medio de su nombre la cadena 'gl', luego ejecuta strip que es un programa que se encarga de quitar simbolos no usados que quedaron remanentes de la compilacion, texto en definitiva. strip se encarga de optimizar en tamaño nuestros binarios.
Cerramos el ciclo for con 'done'.
Luego creamos enlaces blandos a las librerias recien creadas en la compilacion.

Ahora nos encargaremos de crear el modulo necesario para el kernel, el cual se insertara al vuelo una vez ejecutado el 'init' para poder tener acceso al framebuffer de la placa de video, en definitiva, para poder usar las librerias SVGAlib:

Copiamos las librerias optimizadas en tamaño a la locacion final:

```
$ mkdir -p $HOME/rfs/usr/lib
$ cp -a libvga* $HOME/rfs/usr/lib
```

Es hora de compilar el modulo para el Kernel (ahora vemos porque es necesaria la caracteristica de aceptar modulos en el Kernel, aunque la construccion haya sido monolitica):

```
$ cd ../kernel/svgalib_helper
$ make KDIR="$KERNEL"
$ mkdir -p $HOME/rfs/lib/modules
$ cp svgalib_helper.o $HOME/rfs/lib/modules
```

Ahora, vamos a crear el archivo de configuracion de svgalib dentro de etc/ de nuestro nuevo RFS:

```
$ cd ~
$ mkdir -p $HOME/rfs/etc/vga/lib/
$touch $HOME/rfs/etc/vga/lib/libvga.config
$ vim $HOME/rfs/etc/vga/lib/libvga.conf
```

```
mouse PS2
mdev /dev/misc/psaux
```

(esc):wq

```
$
```

Configuracion e instalacion de svncviewer

Vanos a copiar en $HOME los tarballs de tigthvnc y svncviewer, tambien asumimos que el directorio donde se encuentran las fuentes de SVGAlib se encuentran tambien en $HOME:

```
$ cd ~
$ cp -v sources/svncviewer-0.1.1.tar.gz .
$ cp -v sources/tigthvnc-1.2.8_unixsrc.tar.bz2 .
```

Con esto hemos copiado en $HOME los tarballs.

****NOTA: no omitir el punto, ya que este indica al interprete que debe copiarlo en el directorio donde actualmente estamos parados, en este caso $HOME****

Ahora en $HOME tenemos el directorio de las fuentes de svgalib y los tarballs de svncviewer y tigthvnc.

Compilando svncviewer

Como dice Csaba Henk, para obtener la compilacion de svncviewer, necesitamos la libreria 'libvncauth.a' que se encuentra dentro de la distribucion de tigthvnc. Esto realmente complica las cosas, porque en definitiva debemos compilar un fuente dentro de otro fuente.
Para esto, tenemos una herramienta llamada 'xmkmf' la cual ayudara en el trabajo de compilar svncviewer y libvncauth.a, creando archivos 'Makefile' para luego si compilar con make.

Encima, 'xmkmf' no funciona con uClibC, asi que tendremos que usar xmkmf de cara a glibc.

Entonces tenemos que compilar ambos libvncauth.a y svnviewer de la siguiente manera:

•Ejecutamos xmkmf (imake) usando glibc
•Ejecutamos make con opciones especificas para usar uclibc, las fuentes de SVGAlib y agregamos optimizacion en tamaño.

En la practica esto significa lo siguiente: primero, definimos una variable la cual alojara el path (o ruta) relativo que vamos a usar en el proceso de compilacion cuando nos encontremos dentro del directorio donde estan las fuentes de svncviewer, apuntando hacia el directorio donde estan las fuentes de SVGAlib, declaramos la variable:

```
$ export SVGADIR=../../svgalib-1.9.18
```

Ahora tenemos que introducirle una opcion a svncviewer la cual fuerza al mismo a usar un modo SVGA (super vga) determinado, dado por SVGAlib al momento de detectar la placa de video y arrancar svncviewer.

Esto ocurre cuando SVGAlib autodetecta la placa de video y hace su propio 'vgatest' con la opcion de autodeteccion que definimos con anterioridad en el Makefile de SVGAlib. El metodo utilizado se puede consultar en la documentacion que provee SVGAlib.
Aqui vamos:

```
$ bzcat tightvnc-1.2.*_unixsrc.tar.bz2 | tar xv
$ cd vnc_unixsrc
$ tar xvzf ../svncviewer-0.1.1.tar.gz
$ cd libvncauth
```

Como vemos aqui, descomprimimos usando bzcat, que descomprime los archivos y los manda a la salida standard, luego con la tuberia hacia tar terminamos de desempaquetar todos los archivos. Ahora entramos al directorio descomprimido de tigthvnc y desde alli ejecutamos el 'tar' para descomprimir a svncviewer en el lugar donde estamos parados, tomando como referencia el tarball que esta en el nivel superior de directorio ($HOME), por ultimo, ingresamos al directorio libvncauth (no confundir con libvncauth.a).

Es tiempo de la compilacion y para eso, antes que nada debemos dejar a cero la variable que nos indica que libreria C standard debemos usar. Esto lo realizamos con el comando 'unset'.

Como recordaras, en nuestro .bash_profile, habiamos declarado la variable CC (la que determina nuestro compilador C standard como 'export CC=i386-uclibc-gcc'.
Comienza la compilacion:

```
$ (unset CC; xmkmf)
$ make "CC=i386-uclibc-gcc" "CDEBUGFLAGS = $CFLAGS -fno-strength-
reduce" \
 "AR = i386-uclibc-ar clq"
$ cd ../svncviewer
```

Lo que hemos hecho aqui es dejar a cero CC y ejecutar xmkmf, el cual creara un archivo 'imake', luego en las opciones de make declaramos que vamos a usar uClibC y que necesitamos que sea lo mas chico posible, para eso la variable CDEBUGFLAGS contiene otra variable, la variable CFLAGS, la cual indica a uClibC que utilice el metodo -Os de optimizacion en tamaño ademas del argumento -fno-strength-reduce que se encarga de no permitir ejecutar loops al compilador que a veces por tratar de optimizar en tamaño del codigo, devuelve un codigo ejecutable lento, ya que fuerza al mismo compilador a optimizarlo en varias pasadas, eliminando variables y simbolos no usados.

****NOTA: Esta implementacion se basa principalmente en la practica, no en una deduccion teorica, ya que quizas se ha probado antes de forma tradicional y ha resultado, seguramente, un ejecutable lento.****

Por ultimo nos alojamos en el directorio svncviewer que se encuentra un nivel arriba de nosotros. Antes de proseguir, debemos cambiar el mapeado del teclado, ya que en la version original de svncviewer, el mapeado esta hecho para teclados de habla inglesa, por suerte existe la solucion al re-mapeo completo y es el archivo que adjunto llamado 'keys-es.h' el cual debe reemplazar al archivo keys.h que se encuentra dentro del directorio donde actualmente estamos parados: svncviewer.

Hacemos una copia de seguridad de keys.h y reemplazamos por el mapeado en castellano latinoamericano:

```
$ mv keys.h keys.h.old
$ cp ../../sources/keys-es.h keys.h
```

Listo, empezamos la compilacion de svncviewer:

```
$ (unset CC; xmkmf)
$ make "CC=i386-uclibc-gcc" "CDEBUGFLAGS = $CFLAGS -fno-strength-
reduce" \
 "INCLUDES = -I../include -I. -I$SVGADIR/include -I$SVGADIR/gl" \
 "SVGALIB = $SVGADIR/sharedlib/libvga.so
$SVGADIR/sharedlib/libvgagl.so"
$ strip svncviewer
$ mkdir -p $HOME/rfs/usr/bin
$ cp svncviewer $HOME/rfs/usr/bin
```

Por ultimo, dejamos a cero CC nuevamente y ejecutamos xmkmf. Luego ejecutamos 'make' con las opciones anteriormente explicadas, pero agregandole las cabeceras de SVGAlib con la variable INCLUDES. Ahora optimizamos codigo con strip (hasta 30KB podemos eliminar), creamos el directorio donde sera alojado svncviewer (dentro de rfs/) y lo copiamos dentro.
Ahora lo ultimo que nos queda, es poblar a 'rcS' en nuestro nuevo RFS.

Como funciona init en este sistema

Una vez que el kernel arranca, luego de descomprimirse y desempaquetarse, busca su RFS, el cual debe tener un proceso padre llamado init, que en este caso es '@linuxrc' el cual apunta hacia busybox, linuxrc sabe que tiene que ejecutar rcS, el cual levantara los servicios necesarios para hacer funcionar nuestro sistema.
Como sabemos, rcS ya ha sido construido anteriormente, ahora solo falta poblarlo:

```
$ cd ~
$ vim $HOME/rfs/etc/init.d/rcS

#!/bin/sh
insmod /lib/modules/svgalib_helper.o
ifconfig eth0 192.168.0.100 broadcast 192.168.0.255 netmask
255.255.255.0
svncviewer 192.168.0.254:1
```

(esc):wq

Aqui le indicamos a init que carge al vuelo el modulo svgalib_helper.o con insmod, al cual tambien tenemos que pasarle el path completo de donde se encuentra el modulo.
Luego inicializamos la placa de red eth0 (asi se llama la unica que tenemos en nuestra maquina cliente) con la direccion IP 192.168.0.100, ajustamos su broadcast y su mascara tambien.

Por ultimo llamamos al VNC SERVER que se encuentra escuchando permanentemente en la maquina SERVIDOR 192.168.0.254. El ':1' es el modo de video que vamos a usar, en este caso 800*600. Esto lo vamos a ver mas adelante cuando configuremos el servidor.

Hemos concluido con la configuracion y construccion de todo el sistema, es hora de insertar nuestra implementacion dentro de un medio fisico.

Hacia la MICRODISTRO en diskette

¡Bien! Llego la hora de plasmar todo lo hecho en un disco de arranque, guardando dentro de el, al kernel junto a su RFS.
Lo primero que vamos a hacer es formatear nuestro diskette para eliminar el error fisico, insertamos en la diskettera el diskette y tipeamos:

```
$ su
# fdformat /dev/fd0
```

La opcion 'fdformat' se encargara de formatear a bajo nivel nuestro floppy y en formato ext2 para nuestro uso.

Es tiempo de jugar

Lo primero que vamos a hacer es anotar el tamaño fisico de nuestro nuevo root file system alojado en rfs:

```
# cd/home/utc/rfs
# du -h

4,0K ./dev
4,0K ./proc
8,0K ./etc/init.d
8,0K ./etc/vga
20K  ./etc
4,0K ./sbin
108K ./bin
20K  ./lib/modules
288K ./lib
4,0K ./mnt
36K  ./usr/bin
4,0K ./usr/sbin
196K ./usr/lib
240K ./usr
4,0K ./var/log
8,0K ./var
684K .
```

En mi contenedor rfs puedo ver que el total es 684KB, anotamos este resultado.
Ahora vamos a medir el tamaño exacto del kernel, generalmente, con la opcion -h de ls es suficiente, ya que nos devuelve el valor en KB del archivo junto con la suma de resto que ahora paso a explicar, pero si este falla en la lectura y no contempla el resto, debemos hacer una conversion byte>kilobyte, la cual se realiza dividiendo por 1024 el valor de bytes de nuestra imagen de kernel (en este caso bzImage), y si nos queda un resto le sumamos 1 (uno) al tamaño total del archivo en KB.
Tambien quiero recordar que la division por 1024 no es arbitraria, ya que como sabes, la unidad byte esta expresada con notacion OCTAL (0-7), lo cual en notacion decimal lo que es 1000, en octal

es 1024, ademas como sabemos Kilo es multiplo de la unidad que en este caso es el byte.

Para ser mas claros veamos mi kernel:

```
# cd..
# ls -l bzImage
-rw-r--r--  1 root root 520968 abr  7 20:15 bzImage
```

Vemos que el tamaño es de 520968 bytes, bien, ahora vamos a dividir este valor en 1024:

```
520968/1024 = 508,7578125
```

Esto significa 508KB con un resto de 0.7578125 KB, entonces a nuestros 508KB le sumamos 1 para que la copia reserve 1 bloque entero para este 'casi kilobyte'.

¿Pero por que hacemos esto?, la respuesta mas simple es que el gestor de copia a bajo nivel (sin necesidad de montar, lo hace directamente en el dispositivo fisico) que vamos a usar es 'dd'. A dd debemos especificarle el tamaño de cada bloque a copiar, que en este caso utilizaremos la igualdad 1 bloque = 1 kilobyte y la cantidad de bloques a copiar, en este caso 508+1.

Si nosotros le decimos a dd que copie solo 508 bloques de 1KB cada uno, dd hara especificamente lo que nosotros le digamos y copiara hasta el bloque 508 del kernel, dejando fuera el resto que en este caso vimos que es de 0.75KB, que como sabemos, es parte de la imagen en si misma tambien.

Para eso le sumamos 1 a los 508 bloques para ahora si contener dentro de esos 1024 bytes la informacion contenida en ese 0.75KB (aproximadamente 732 bytes).

Ahora otra cosa que debemos tener en cuenta es el tamaño final del kernel junto a nuestro nuevo RFS. La suma de ellos no debe superar por ningun motivo el tamaño maximo permitido de 1440 bloques (¡adivinaste! Esos '1440' bloques significan el tamaño de nuestro diskette). Vamos a sumar mi RFS y mi kernel, veamos:

Mi RFS segun 'du -h': 684KB

Mi imagen bzImage segun mis calculos: 509KB

Sumamos RFS y mi imagen del kernel:

RFS+bzImage = 1193

¡Genial! Todavia tenemos resto, como podemos ver, el total es igual a 1193KB y todavia nos queda resto dentro del diskette.
Ahora solo nos queda meter en nuestro diskette el kernel y el RFS.

Metiendo todo en el diskette

El metodo que vamos a usar es el de utilizar imagenes comprimidas tanto del kernel como nuestro RFS, para que luego, una vez cargadas, se desempaqueten y descompriman en memoria, a esto le llamamos un 'disco RAM', la maquina tiene cargada en su memoria fisica al kernel y a todo su sistema de archivos, esto tiene la ventaja de no tener que usar disco duro pór ejemplo, pero tiene la

desventaja de que el espacio reservado para este tipo de dispositivo de bloque llamado disco RAM, es estatico y una vez reservada la memoria, esta no es liberada bajo ningun motivo, hasta que se oprime el boton de reset o se apaga la maquina, perdiendo asi todos los datos alojados en la memoria fisica de la maquina, por supuesto, aqui no sufrimos ante este hecho, ya que no debemos guardar permanentemente ningun dato.

***NOTA: Sobre el hecho de que la memoria es reservada por el kernel para su disco RAM el cual alojara finalmente a su propio RFS, es un problema que tuve que enfrentar hasta poder lograr que el sistema me funcione estable con tan poca memoria fisica.
Mi consejo es que no reserves memoria fisica extra para la imagen de tu nuevo RFS, ya que esa memoria faltara luego cuando por ejemplo quieras ejecutar remotamente un explorador como FIREFOX y luego navegar por internet.

Tuve el problema que la conexion del cliente vnc se perdia y la maquina cliente se congelaba cuando Firefox intentaba navegar por paginas que contenian algo mas que texto plano. Esto, obviamente, estaba causado por falta de memoria, recuerda que aqui aprovechamos hasta el ultimo KB dentro de los recursos y conociemientos basicos que podemos tener. Como siempre digo, de seguro podemos achicar mas todavia tanto al kernel como al nuevo RFS, pero esto lo dejare para una futura revision.***

Sabemos que el tamaño de nuestro filesystem es (tomo como ejemplo ilustrativo mi RFS) de 684KB, bien, entonces redondearemos a 700KB el tamaño de nuestro disco RAM.
Vamos a crear un dispositivo de bloque llamado rootfs como archivo de salida, lleno de ceros, los cuales los tomamos del dispositivo especifico que los genera: '/dev/zero' como archivo de entrada, con tamaño de bloque igual a 1KB y un total de 700 bloques:

```
# dd if=/dev/zero of=rootfs bs=1k count=700
```

Ahora formateamos este dispositivo el cual no posee en realidad ningun formato especifico, con la aplicacion 'mke2fs', en el cual agregamos el argumento -F para formatear, -m 0 para indicarle que no reserve memoria para el usuario root y un numero -N de 400 inodos (los inodos son identificadores unicos de cada archivo, en realidad el inodo es un numero que forma parte de un gran indice de numeros, el cual por medio del nro. identificador de inodo sabe a que parte del dispositivo de bloque debe leer, por supuesto, cada archivo tiene (en realidad debe tener) su nro. de inodo identificatorio. Yo he usado un valor de 400 para no dejar corto al kernel a la hora de crear archivos al vuelo cuando sea necesario. Para saber cuantos inodos tenemos, contemos cuantos archivos, enlaces y directorios tenemos (en mi caso 125)):

```
# mke2fs -F -m 0 -N 400 rootfs
```

Ahora, ya que hemos formateado nuestro flamante sistema de archivos llamado rootfs, es hora de montarlo en un directorio que en este caso crearemos con el nombre de 'temporal'. El modo de montaje es el modo 'loop', el cual nos permite montar al vuelo una imagen de disco (para mas info ver 'man mount'):

```
# mkdir -p temporal
# mount rootfs -o loop temporal
```

Dentro del directorio 'temporal' podemos ver que se encuentra otro directorio llamado 'lost+found', este directorio es usado unicamente por fsck, el programa para reparar archivos de linux. Lo

eliminamos ya que no vamos a usar fsck nunca, ya que no lo hemos incluido y ademas porque estamos trabajando en un diskette el cual posee un tiempo de vida muy limitado.

```
# rmdir temporal/lost+found
```

Ahora viene lo mejor: copiamos todo el contenido de rfs/ dentro de temporal/

```
# cp -a rfs/* temporal
```

Ahora, todo el contenido de temporal/ pertenecera a root:

```
# chown -R 0:0 temporal/*
```

Desmontamos 'temporal'

```
# umount temporal
```

Finalmente convertimos este dispositivo de bloque lleno de informacion a una imagen comprimida de disco RAM, utilizando el poder de las tuberias o pipes. Aqui le decimos a dd que tome a rootfs y lo vaya entregando por su salida standard, en bloques de 1KB, a la entrada standard de gzip, el cual comprime en su rate maximo con -v9, a su vez, la salida standard de gzip es redireccionada hacia un archivo nuevo llamado 'root.gz'

```
# dd if=rootfs bs=1k | gzip -v9 > rootfs.gz
```

Instalando el KERNEL y el disco RAM dentro del diskette

Como ultimo paso, debemos copiar el kernel junto al RFS dentro del diskette para que podamos tener un sistema operativo completo.
Como te dije antes, la serie 2.4.* del kernel te posibilita que puedas cargar tu RFS directamente sin necesidad de un bootloader, para ello debemos activar algunas banderas dentro de la imagen del kernel, pero antes lo instalamos:

```
# dd if=bzImage of=/dev/fd0 bs=1k
```

Aqui hemos copiado la imagen del kernel directamente sobre el diskette (/dev/fd0)

```
# rdev /dev/fd0 /dev/fd0
```

Aqui la cosa se pone interesante, ya que 'rdev' se usa para modificar parametros dentro del kernel.

El primer 'rdev' le dice al kernel que establezca al dispositivo /dev/fd0 como su RFS y que solo se le tiene permitido montar /dev/fd0.

```
# rdev -R /dev/fd0 0
```

La opcion -R establece que las opciones del sistema raiz esten a 0 (cero).

```
# rdev -r /dev/fd0 16893
```

La opcion -r establece el tamaño del disco RAM, pero cuidado, que el numero que le precede no es un numero que indique tamaño, sino un 'numero magico'.

En linux es comun encontar este tipo de expresiones, que en este caso, este numero es la suma resultante de un bit bandera dentro del kernel de un conjunto de 16 bits banderas (o comunmente llamados flags), el cual esta alojado en la posicion 14.

Los flags se utilizan, como su nombre lo sugiere, para indicar algo. Como los posibles valores son 0 y 1, generalmente se usan para desactivar alguna caracteristica con '0' o activar esa caracteristica con '1'.

en este caso nosotros activamos el bit 14 como marca para indicarle al kernel que debemos cargar una imagen de disco RAM, pero todavia debemos pasarle el tamaño de su propia imagen, el cual queda contenido dentro de los bits 0 al 10.

Entonces, tenemos que el bit 14 representa la accion de cargar una imagen RAM, bien, el valor real del bit 14 es simplemete de 2^14, el cual es igual a 16384.
Ahora sumamos a este numero el valor del tamaño de nuestro KERNEL y obtenemos el 'numero magico' 16893:

```
16384+509 = 16893
```

El grafico para entender esto seria algo asi:

bits 0 al 10	***bits 11-12-13***	***bit 14 (0 o 1)***	**bit15**
tamaño de bzImage	estos bits no se usan	flag RAMDISK (off/on)	????????

¡PERO TODAVIA HAY MAS!

Como podemos ver, todavia nos queda 1 bit mas, este bit, se utiliza para decirle al kernel, que en vez de cargar un disco RAM que contenga un RFS en el mismo diskette, pida un nuevo diskette con un RFS completo. Esto es tremendamente util cuando no nos importa el tamaño y asi de esta forma, usamos un diskette para alojar al kernel y otro para el RFS.
Para esto, debemos poner a 0 el bit 14 y a 1 el bit 15.

Cuando ponemos a 0 el bit 14 conjuntamente con el bit 15 a 1, el kernel interpreta que:
primero, el disco RAM se encuentra en otro diskette y que al estar en 0 el bit 14, le indica que cargue el RFS a partir del bloque 0 del diskette donde esta alojado.
Ahora, para obtener el 'numero magico' vamos como en el caso anterior a obtener el valor decimal de 2^15 que es igual a 32768, el cual triquiñuela aparte, lo sumamos a 16384, obteniendo asi 49152:

```
32768+16384 = 49152
```

De esta forma, este nuevo valor le indica al nucleo que pida un nuevo diskette y cargue desde el bloque 0 una imagen de disco RAM. Pero esto ultimo no lo vamos a usar en esta implementacion, solo es a titulo informativo.

```
# dd if=rootfs.gz of=/dev/fd0 bs=1k seek=509
```

Por ultimo, copiamos al diskette la imagen de disco RAM 'rootfs.gz' en bloques de 1KB, advirtiendole con 'seek' que el tamaño de bzImage es de 509 bloques, de esta forma, comenzara a copiar en el bloque inmediatamente posterior al bloque 509.

¡Exito!
Hemos concluido finalmente la ardua tarea de crear una microdistribucion GNU/LINUX que pueda arrancar en una maquina tipo i486 con solo 4MB RAM, un chipset de video tipo Cirrus Logic, Bus de teclado y raton tipo PS/2 y placa de red NE2000 ISA.
Podemos probar nuestro diskette insertandolo en la maquina cliente y encendiendola. Veremos que primero se carga el kernel, luego se descomprime, despues comienza a autodetectar nuestro hardware, mas tarde se detiene a leer el disco RAM que se encuentra en el bloque inmediatemente posterior al ultimo especificado por el kernel. Luego de esto vemos como se ejecuta BUSYBOX llamando a la libreria SVGAlib con el modulo que se inserta en el nucleo al vuelo, despues de esto el sistema arranca svncviewer, este autodetecta la placa de video y setea los modos de video disponibles para esta placa, luego trata de conectarse a un servidor grafico, el cual todavia no hemos preparado.
En el proximo capitulo, vamos a ver como dejar a punto un servidor VNC para concluir nuestra aventura.

CAPITULO 4: Implementando nuestro Servidor UTC

Implementando nuestro SERVIDOR VNC por demanda

La filosofia de trabajo de este tipo de soluciones tecnologicas se denomina 'Terminal Server' ya que se utilizan terminales bobas o clientes finos para acceder a un Server grafico.
Una de las implementaciones mas conocidas en el mundo de los Terminal/Server es el proyecto LTSP (Linux Terminal Server Project) de Jim McQuillan, el cual comenzo como una implementacion enfocada a dar una solucion a una empresa y termino siendo uno de los proyectos mas importantes actualmente en el mundo GNU/LINUX.

No me voy a centrar de forma directa en la implementacion de un LTSP, pero a ciencia cierta, voy a abordar la configuracion de un servidor LTSP de forma indirecta, ya que se necesitan 5 (cinco) servicios: xinetd, XDMCP, tftp, nfs y dhcp, justamente los 5 servicios basicos tambien de un entorno LTSP, pero la diferencia radica en que en realidad no vamos a implementar un server LTSP, sino usar las herramientas necesarias para ofrecer el servicio de cliente VNC por demanda que por uso standarizado tambien se usan en LTSP, TcOs, ThinStation, etc.

¿Pero, por que tantos servicios?, ¿que hace cada uno?

Bien, xinetd (e**x**tended **In**ternet s**e**rvices **d**aemon) es un servicio que se encarga de escuchar constantemente una peticion remota, la cual se accede mediante un puerto especifico, en este caso, se encarga de manejar a 2 (dos) servicios cruciales: tftp y vncserver. Cuando es realizada una peticion al puerto correspondiente a tftp o a vncserver, xinetd libera el servicio.

XDMCP (**X D**isplay **M**anager **C**ontrol **P**rotocol) es un manejador de sesiones graficas remotas, pero antes de entender como funciona XDMCP, estaria bueno repasar el concepto de Xwindow:

Entendiendo Xwindow

Xwindow es un sistema de entorno grafico para maquinas tipo UNIX/LINUX, el cual funciona de modo SERVER, ofreciendo los servicios de dibujo en pantalla, posicionamiento de raton, acceso a teclado, mientras que las aplicaciones en si, trabajan como clientes de este servidor. Bajo este concepto podemos ejecutar aplicaciones de forma remota, mientras que el server X se encuentra local. Pero hay que tener en cuenta que X no es un gestor de ventanas, ya que no las genera, para eso estan los gestores tales como xdm, gdm o kdm.
Ahora, la forma mas simple de entender que es XDMCP, es verlo como un manejador de peticiones graficas remotas al SERVER X que se encuentra en una maquina HOST.

Entendiendo TFTP

El servicio **tftp** (**t**rivial **f**ile **t**ransfer **p**rotocol) se utiliza para transportar archivos cuando todavia no existe la capa de aplicacion de usuario en el hardware, justamente por eso se llama trivial, ya que no posee ninguna politica de seguridad, ni siquiera usuario o contraseña. En este caso lo vamos a usar para transportar el kernel que hemos construido desde el servidor hacia el cliente, cuando obviamente todavia no poseemos ningun tipo de sistema cargado en nuestra 486, solo una rutina de arranque que mas adelante veremos.

Entendiendo NFS

El servidor **NFS** (**N**etwork **F**ile **S**ystem) tambien es conocido como un protocolo de red homonimo. La ventaja de este protocolo nativo de UNIX/LINUX es que es transparente y simple de utilizar, permitiendo acceder a recursos remotos como si se trataran de recursos locales. Este protocolo se

divide en dos partes, el 'servidor nfs' y el 'cliente nfs'. Una de las caracteristicas principales de nfs es que funciona de forma sincrona, esto significa, que no libera al servicio hasta que no haya completado la tarea encomendada, esto preserva la integridad de los datos. En resumen: nfs es un protocolo eficiente de acceso a recursos nativo de linux.

Entendiendo DHCP

El servidor **DHCP** (**D**ynamic **H**ost **C**onfiguration **P**rotocol) (Protocolo de Configuracion de Hosts Dinamicos) es un protocolo de comunicacion basado en una parte servidor y otra cliente.
Basicamente dhcp es un servcio de asignacion de identidad dinamico a los nodos (todavia no tienen identificacion, por eso nodos) que lo requieran via TCP/IP.
El principio de funcionamiento del server dhcp es simple: dhcp posee una lista de posibles asignaciones de identidad, ya sea de forma estatica (mediante control de las direcciones MAC de las placas de red de los nodos, los cuales poseen un cliente dhcp) o de forma dinamica (o 'scope', que en realidad se trata de un rango de direcciones IP el cual sera entregada segun vayan generandose las peticiones remotas desde los nodos, los cuales poseen un cliente dhcp).
En detalle, se suceden 5 pasos para la configuracion del nodo, el cual se convierte en un host con identificacion propia:

1) DHCPdiscovery:

El cliente dhcp envia un paquete denominado DHCPDISCOVER, el cual contiene los datos IP 0.0.0.0 y mascara 255.255.255.255, que en definitiva, significa una llamada 'broadcast' a toda la red, pidiendo identificacion. Junto con DHCPDISCOVER viaja otro paquete CHADDR, que contiene la direccion MAC o identificador unico (**:**:**:**:**:**hex) de la placa de red.

2) DHCPoffer:

Sucede ahora que el servidor dhcp reecibe la peticion y determina la configuracion de futuro host leyendo el paquete CHADDR y comparandolo con una lista de posibles hosts identificados por la direccion MAC. Ahora el servidor especifica una direccion IP y la envia en un paquete llamado YIADDR.

3) DHCPrequest:

El cliente toma los datos de DHCPOFFER, los lee y una vez mas, el cliente pide especificamente 'esa' identificacion que acaba de recibir via el paquete YIADDR.

4) DHCPAcknowledge:

El servidor acaba de recibir nuevamente la peticion mediante DHCPREQUEST y de esta forma concluye la configuracion del cliente, enviandole el paquete DHCPACK con la configuracion final de la IP, mascara de red, gateway y demas datos especificos de red que seran usados por el cliente para identificarse propiamente en la red.

5) DHCPRelease:

El cliente, ahora host, envia una señal, no de broadcast tipo multicast, sino direccionada, tipo unicast al server dhcp, liberando su nueva direccion IP. De esta forma concluye la configuracion de host en la maquina cliente.

Por ultimo, **VNCSERVER** es nuestro servidor de VNC (**V**irtual **N**etworking **C**omputation o en Castellano: computacion en red virtual) el cual nos permite abrir una sesion grafica de forma remota desde un cliente VNC. Las caracteristicas mas importantes de este sistema es que es multiplataforma, ya que no es necesario que las dos maqunas, el server y el cliente, posean el

mismo sistema operativo, mientras cada una de ellas posean su servidor y cliente VNC. Otra importante ventaja, es que lo que el cliente emite solo eventos de raton y teclado y solo recibe una pintura refrescada 50 veces por segundo de lo que pasa en la sesion abierta en el servidor. Podemos compartir una misma sesion en todas las maquinas, esto es especialmente util cuando necesitamos enseñar de forma interactiva a un curso completo.
El principio de funcionamiento de VNC es muy simple: se basa en el principio de video FRAMEBUFFER remoto, o sea, enviar informacion por la red, directamente a un dispositivo de video con FRAMEBUFFER. El framebuffer es basicamente una de las capas mas bajas de una placa de video. Tenemos entonces que el framebuffer es un dispositivo virtual que lee cada bit de la memoria de video y la representa exactamente en pantalla. De esta forma es como trabaja VNC: el servidor envia toda la informacion directamente a la memoria de video de la placa, y mediante el cliente VNC esta es representada en pantalla gracias al framebuffer que es leido una vez mas por el cliente VNC.

Ya he concluido con las descripciones basicas de los servicios asociados a esta implementacion, ahora es momento de configurar nuestro servidor.

Preparando nuestro SERVIDOR

El hecho es simple: sin un servidor que nos sirva un entorno grafico, aplicaciones, acceso a internet, en definitiva un TERMINAL SERVER, no podemos hacer mucho con nuestra microdistribucion, ya que ha sido construida especificamente para el uso de maquinas con escasos recursos, en los cuales es imposible incluir un servidor grafico local y aplicaciones.
Nuestro Servidor se encargara de ofrecer todas las utilidades necesarias para el usuario que se presente frente a la maquina cliente.

Dos caminos, una eleccion

Te voy a presentar dos caminos en la configuracion del SERVER, te toca elegir cual quieres usar. Los posibles caminos son el camino facil y el camino dificil, que por supuesto comienzo a describir.

El camino facil

La forma mas facil y rapida de poner en marcha un entorno Terminal/Server con caracteristicas de servidor VNC por demanda, es utilizando una distribucion Linux que posea todos los servicios anteriormente mencionados, preconfigurados.
Aqui yo recomiendo la linea clones de Red Hat: Fedora y CentOs en sus versiones K12LTSP.
Las implementaciones han sido probadas con exito en las siguientes versiones:

FEDORA K12LTSP-3
FEDORA K12LTSP-4
FEDORA K12LTSP-5
FEDORA K12LTSP-6
CENTOS K12LTSP-EL5

La eleccion de por que K12LTSP, es simplemente por la incorporacion de software educativo, ya instalado, para niños de 3 a 12 años, lo cual de esta forma, nos evitamos tener que ponernos a investigar que software deberiamos usar para este de perfil de implementacion y porque la configuracion del servidor VNC ya viene configurada de tal forma que insertamos el diskette en la maquina cliente, la encendemos y solo tenemos que esperar despues de casi un minuto de espera una pantalla grafica de login.

Lamentablemente, el desarrollador de la version K12ltsp, abandono Fedora en la version 6 y migro a CentOs EL5, agregando los paquetes K12LTSP, resultando ahora CentOs K12LTSP-EL5.

****NOTA: La instalacion de un sistema K12LTSP solo tarda 20 minutos, una vez instalado el servidor y asumiendo que ya estan en red tanto la maquina cliente como el Server, solo debemos inicializar la maquina cliente con el diskette dentro, esperar cerca de 1 (uno) minuto y veremos la resultante pantalla grafica de login:****

Una cosa mas a tener en cuenta: recomiendo dejar los valores predeterminados de configuracion de red al momento de la instalacion, el cual advierte que la direccion IP del SERVER sera 192.1680.254 y si por esas casualidades no te has dado cuenta, es justamente la direccion IP que declaramos en el cliente VNC, svncviewer, dentro de 'rcS' alojado en la microdistro.

El camino dificil

Vamos de lleno a configurar manualmente los servicios necesarios para que nuestra microdistro funcione correctamente.
De esta forma logramos que la implementacion sea independiente de la distribucion que usemos.

Las distribuciones en donde he probado con exito han sido:

Fedora 2-3-4-5-6-7-8-10
CentOs 4.* y 5.*
Ubuntu series 6.*
EdUbuntu series 6.*
Xubuntu series 6.*
OpenSuse 10.*

Intentare explicar paso a paso la instalacion desde cero de un servidor tipo Terminal/Server, el cual nos llevara seguramente de 2 a 3 horas como minimo de instalacion, configuracion y puesta en marcha completa.

Otra cosa que quiero aclarar es que basicamente, con estas instrucciones es posible implementar esta solucion usando CUALQUIER distribucion GNU/LINUX, asi que no tengas miedo de experimentar, mas si ya has llegado hasta este punto.

¡Quiero mi implementacion ya!

¡Ya tenemos listo nuestro diskette de arranque, pero todavia no podemos probar nuestro trabajo!, bien en esta primera parte de la configuracion del servidor, veremos como dejar listos los dos servicios basicos para poder hacer uso total de la microdistribucion alojada en diskette. Estos dos servicios son muy faciles de configurar y muy intuitivos, el primero es xinetd, el super demonio de internet y el segundo es XDMCP, el manager de conexiones externas al Xserver de nuestro servidor. Con esto ya tenemos asegurado el funcionamiento exitoso de nuestra implementacion en diskette. Las siguientes instrucciones sirven para cualquier distribucion de GNU/LINUX, de todas formas, paso a explicar una a una las implementaciones que con exito he conseguido.

Implementando un Servidor UTC en Fedora 10

Esta distribucion es la mas actualizada principalmente a la hora de soportar hardware, es la mas innovadora a la hora de incluir caracteristicas y tiene una comunidad enorme en todo el mundo, es rapida, muy estable y bonita.

¡Si! ¡Fedora 10! para los que creian que era imposible implementar algo como esto en esta distribucion (me incluyo, ya que al principio no pude, pero la comunidad de Fedora se puso manos a la obra y soluciono un BUG heredado que traia GDM por defecto en la version 2.24).

****NOTA: En realidad muchos dicen que no fue un bug sino una decision de los desarrolladores. El problema en si es que la aplicacion 'gdmsetup', la famosa 'ventana de entrada' que se veia con Gnome en Sistema > Administracion habia desaparecido y no habia forma de setear correctamente XDMCP, porque segun los responsables, estaban en desacuerdo con facilitar acceso remoto grafico multiple en la version 2.24 de Gnome, cosa que se revirtio en la nueva version 2.24.1-4 y aparecera nuevamente en la futura version 2.26. (al momento de escribir esto, estoy en JUL 2009)****

La implementacion en Fedora 10 es muy sencilla y solo ocupa unos cuantos pasos a seguir. Lo primero que haremos sera una instalacion por defecto de escritorio, elegiremos cuando se nos pida solo la opcion de 'ofimatica', de esta forma, luego al instalar el resto de los servicios, tendremos el total control sobre la distribucion.

En la instalacion cuando te pregunte por la direccion IP que queremos que tenga tu maquina, le especificas que sea fija y con los siguientes datos:

```
Static or Dhcp:     Static
IP-ADDRESS:         192.168.0.254
NETMASK:            255.255.255.0
GATEWAY:            192.168.0.254 # o donde apunte la pasarela tuya
Ipv6:               Disabled
```

Una vez que hayas terminado de instalar tu nuevo y flamante Fedora 10, es hora de actualizarlo con unicamente los repositorios OFICIALES. Esto te demorara un tiempo aproximado entre 1 o 2 horas dependiendo de tu velocidad de conexion. Estoy hablando de aproximadamente 600MB de paquetes. Esto lamentablemente es asi, ya que estamos al final del tiempo de vida de Fedora 10, ya que en pocos dias esta por salir la version 11 y por supuesto, la distribucion ha ido renovando sus paquetes.

****NOTA: espero con ansias en la version 11 el nuevo sistema 'Delta' para los paquetes rpm, que consiste en solo actualizar el codigo que difiere y no todo el paquete, esto en otras palabras significa que si antes necesitabas completar una actualizacion de 600MB, ahora solo necesitas bajar 60MB. ¡Todo un logro de la comunidad Fedora!.****

Para actualizar nuestro Fedora 10:

```
# yum update
```

O podrias probar esta forma, que te ahorrara un 95% de tamaño y espera, la cual he probado despues de destrozar mi sistema Fedora 10 mezclando paquetes de CentOs 5.* y Fedora 11 en version beta:

```
# yum update gdm
```

Luego, si tienes tiempo y ganas, lo dejas actualizando como se debe a tu sistema una noche de estas. Ahora comprobamos que la version de 'gdm' es 2.24.1-4:

```
# rpm -q gdm
gdm-2.24.1-4.fc10.i386
```

Habilitando XDMCP

Es hora de modificar un par de parametros en /etc/gdm/custom.conf para habilitar XDMCP, agregando a custom.conf los cambios necesarios para habilitar el acceso remoto:

```
# vim /etc/gdm/custom.conf

# GDM configuration storage

[daemon]
RemoteGreeter=/usr/libexec/gdmgreeter
# Este es el ejecutable que te muestra la pantalla de inicio o
login grafico

[xdmcp]
Enable=true
# Aqui habilito XDMCP

[chooser]

[security]
```

```
[debug]

[greeter]
GlobalFaceDir=/usr/share/pixmaps/
GraphicalTheme=circles
GraphicalThemes=circles
Browser=false
# Estas son las configuraciones esteticas de la pantalla de login,
# la mas destacable
# es 'Browser' que simplemente es el face browser que puedes
# observar en forma
# de una foto o dibujo al lado de cada nombre de usuario en la
# lista de usuarios
# que se te presenta al momento del login grafico.
```

(esc):wq

Con esto es suficiente para habilitar XDMCP y asi conseguir conexiones remotas.
Otros parametros que puedes incluir en la etiqueta [xdmcp] y que son sumamente importantes, especialmente si tienes muchos clientes pidiendo conexion al server, son:

`HonorIndirect`
Valor = booleano (ejemplo: true, false)
Activa la selección XDMCP INDIRECT (ejecución remota de gdmchooser) para terminales X que no suministran su propio explorador de pantallas.

`MaxPending`
Valor = numerico (ejemplo: 4)
Para evitar ataques por denegación de servicio, GDM tiene una cola de tamaño fijo de conexiones pendientes. Sólo pueden iniciarse a la vez MaxPending pantallas.

NOTA: que este parámetro *no* limita el número de pantallas remotas que se puedan gestionar. Sólo limita el número de pantallas iniciando una conexión simultáneamente.

`MaxSessions`
Valor = numerico (ejemplo: 16)
Determina el número máximo de conexiones de pantallas remotas que serán gestionadas simultáneamente.
Ejemplo: el número total de pantallas remotas que puede usar tu servidor.

`MaxWait`
Valor = numerico (ejemplo: 30)
Cuando GDM esté preparado para gestionar una pantalla se envía un paquete ACCEPT a ella conteniendo un id único de sesión que será usado en las conversaciones XDMCP futuras.
GDM entonces colocará el id de sesión en la cola de pendientes esperando a que la pantalla responda con una petición MANAGE.
Si no se recibe ninguna respuesta dentro de MaxWait segundos, GDM declarará la pantalla muerta y la borrará de la cola pendiente liberando el espacio para otras pantallas.

`MaxWaitIndirect`

Valor = numerico (ejemplo: 30)
El parámetro MaxWaitIndirect determina el número máximo de segundos entre el tiempo donde un usuario elije un host y la subsiguiente petición indirecta donde el usuario se conecta al host. Cuando se excede el tiempo establecido, la información acerca del host elegido se olvida y el slot indirecto se libera para otras pantallas. La información se puede olvidar antes si hay más hosts intentando enviar solicitudes que MaxPendingIndirect.

`PingIntervalSeconds`
Valor = numerico (ejemplo: 15)
Intervalo en el ping al servidor X en segundos. Si el servidor X no responde antes de la siguiente vez que se lance un ping a él, la conexión se detiene y la sesión termina. Esta es una combinación de las opciones de XDM PingInterval y PingTimeout, pero en segundos.
Nota que GDM en el pasado usaba una clave de configuración PingInterval que estaba en minutos. Para la mayoría de los propósitos querría que este ajuste fuera menor que un minuto, sin embargo en la mayoría de casos donde se usaría XDMCP (como terminales de laboratorio), un retraso de más de 15 segundos realmente significaría que el terminal se ha apagado o reiniciado y querrías terminar la sesión.

`Port`
Valor = numerico (por defecto: 177)
El número de puerto UDP en el que gdm debería escuchar las peticiones XDMCP. No cambies esto a no ser que sepas lo que estás haciendo.

Comprobando la existencia de xinetd

Simplemente nos fijamos si ya tenemos instalado xinetd haciendo un query a rpm:

```
$ rpm -q xinetd
xinetd-2.3.14-21.fc10.i386
```

Ahora vamos a configurar xinetd, instalar vnc-server y configurar el servicio correspondiente asociado.

Configurando xinetd & sus servicios - Instalando vnc-server

Ahora solo nos queda un par de pasos mas para terminar con la implementacion del servidor. Lo primero que vamos a hacer es aprovechar el legado de la era K12LTSP en los viejos Fedora Core y que fue migrado a Fedora 10 por los responsables actuales encargados de mantener LTSP version 5 y toda la rama de aplicaciones educativas del grupo de instalacion LTSP.

Vamos a chequear que contiene el grupo de instalacion de yum llamado 'LTSP':

```
# yum groupinfo LTSP
```

```
Loaded plugins: refresh-packagekit
Setting up Group Process
Group: LTSP
Description: The Linux Terminal Service Project
Default Packages:
```

MToolsFM caching-nameserver-ltsp dhclient	ltsp_ltspswapd ltsp_ppc ltsp_x_vesa

dhcp	meld
dhcp-ltsp-config	nat
gnome-theme-clearlooks-bigpack	nc
gnome-themes-extras	perl-Authen-SASL
icemc	perl-Convert-ASN1
icewm	perl-IO-Socket-SSL
icewm-l10n	perl-LDAP
icewm-menu-gnome2	perl-NetAddr-IP
icewm-themes	perl-Net_SSLeay
k12ltsp-core	perl-RPM-Specfile
k12ltsp-extras	perl-X11-Protocol
k12ltsp-release	root-desktop-urls
k12ltsp-scripts	schoolbell
k12ltsp-utils	schooltool
kdebase	spheres-and-crystals
lts-eb-bootroms	squidGuard
ltsp-utils	tftp-server
ltsp_config	verynice
ltsp_desktop_reset	vino
ltsp_esound_i386	vnc
ltsp_esound_server	vnc-ltsp-config
ltsp_i386	vnc-server
ltsp_i386-boot	yum
ltsp_i386-config	yumex
ltsp_ltspfs	

Es impresionante la cantidad de paquetes asociados a LTSP, que como veremos mas adelante, tomaremos provecho de algunos de ellos, obteniendo la invalorable ventaja de que ya vienen preconfigurados para un escenario tipo cliente/servidor. Por ahora solo instalaremos el paquete 'vnc-ltsp-config ', el cual trae por defecto una preconfiguracion de vnc-server, xinetd y /etc/services.
Pero antes de dar por sentado nada, te paso a explicar el papel de /etc/services en conjunto a xinetd.
`/etc/services` es un archivo de texto muy extenso el cual posee las declaraciones de puertos y sus servicios asociados. Si te fijas dentro de él te daras cuenta que tiene la siguiente estructura:

Service-name *Nombre del servicio*	port/protocol *puerto/protocolo*	[aliases ...] *[alias...]*	[# comment] *[#comentario]*

VNC escucha a partir del puerto 5900, asi que podemos usar un puerto especifico a partir del 5900 para ofrecer diferentes configuraciones de VNC (5900, 5901, 5902, etc), que en este caso se resumen a diferentes resoluciones de pantalla, pero te lo cuento a modo informativo, ya que solo me interesa declarar una resolucion, la de 800*600 pixeles.
De esta forma xinetd consulta /etc/services cada vez que se inicia para setear cada servicio asociado a su correspondiente puerto de escucha.

Es hora de instalar el paquete:

```
# yum install vnc-ltsp-config
```

Este meta-paquete se encarga una serie de aplicaciones, entre ellas vnc-server, podemos ver que paquetes dependen de el:

```
# yum deplist vnc-ltsp-config

Loaded plugins: refresh-packagekit
Finding dependencies:
```

package: vnc-ltsp-config.noarch 4.0-5 *dependency: config(vnc-ltsp-config) = 4.0-5* provider: vnc-ltsp-config.noarch 4.0-5 *dependency: perl* provider: perl.i386 4:5.8.8-18.el5_3.1 provider: perl.i386 4:5.8.8-10 provider: perl.i386 4:5.8.8-18.el5 provider: perl.i386 4:5.10.0-49.fc10 provider: perl.i386 4:5.10.0-68.fc10 *dependency: xinetd* provider: xinetd.i386 2:2.3.14-10.el5 provider: xinetd.i386 2:2.3.14-21.fc10 *dependency: vnc-server >= 4.0* provider: vnc-server.i386 4.1.2-14.el5_3.1 provider: vnc-server.i386 4.1.2-9.el5 provider: vnc-server.i386 4.1.2-14.el5 provider: vnc-server.i386 4.1.2-35.fc10 provider: vnc-server.i386 4.1.3-1.fc10	*dependency: /bin/sh* provider: bash.i386 3.2-24.el5 provider: bash.i386 3.1-16.1 provider: bash.i386 3.2-29.fc10 provider: bash.i386 3.2-30.fc10 *dependency: chkconfig* provider: chkconfig.i386 1.3.30.1-1 provider: chkconfig.i386 1.3.30.1-2 provider: chkconfig.i386 1.3.38-1

Vamos a averiguar un poco mas de este paquete:

```
# yum provides vnc-ltsp-config

vnc-ltsp-config-4.0-5.fc10.noarch : Easy Enabler of VNC remote LTSP desktops
Repo        : installed
Matched from:
Other       : Provides-match: vnc-ltsp-config
```

¡Todavia hay mas!

```
# rpm -qi vnc-ltsp-config

Name        : vnc-ltsp-config       Relocations: (not relocatable)
Version     : 4.0                   Vendor: Fedora Project
Release     : 5.fc10                Build Date: dom 21 dic 2008 20:23:57 ARST
Install Date      : mié 20 may 2009 20:19:31 ART
Build Host  : xenbuilder2.fedora.redhat.com
Group       : User Interface/X            Source RPM: vnc-ltsp-config-4.0-5.fc10.src.rpm
Size        : 2212                        License: GPLv2+
Signature   : DSA/SHA1, mar 30 dic 2008 04:46:46 ARST, Key ID bf226fcc4ebfc273
Packager    : Fedora Project
Summary     : Easy Enabler of VNC remote LTSP desktops
Description :
Easy Enabler of VNC remote LTSP desktops.  This package contains xinetd.d
```

```
configurations that allow incoming VNC clients to open a new desktop session
with each connection.  Xvnc connects to the local server using XDMCP.
```

Una vez que hayamos terminado de instalarlo, vamos a /etc/xinetd.d y comprobamos que esta presente el archivo 'vncts':

```
# cd /etc/xinetd.d
# ls -l
total 56
-rw-r--r-- 1 root root 1157 sep 18  2008 chargen-dgram
-rw-r--r-- 1 root root 1159 sep 18  2008 chargen-stream
-rw-r--r-- 1 root root  521 may 20 20:11 cvs
-rw-r--r-- 1 root root 1157 sep 18  2008 daytime-dgram
-rw-r--r-- 1 root root 1159 sep 18  2008 daytime-stream
-rw-r--r-- 1 root root 1157 sep 18  2008 discard-dgram
-rw-r--r-- 1 root root 1159 sep 18  2008 discard-stream
-rw-r--r-- 1 root root 1148 sep 18  2008 echo-dgram
-rw-r--r-- 1 root root 1150 sep 18  2008 echo-stream
-rw-r--r-- 1 root root  332 may 20 20:11 rsync
-rw-r--r-- 1 root root 1212 sep 18  2008 tcpmux-server
-rw-r--r-- 1 root root 1149 sep 18  2008 time-dgram
-rw-r--r-- 1 root root 1150 sep 18  2008 time-stream
-rw-r--r-- 1 root root 1890 may 20 20:28 vncts
```

Lo abrimos y miramos dentro de él:

```
# less vncts

# default: off
# To use this remote vnc method:
# * Allow local accesss to Xserver/XDMCP, add to /etc/X11/xdm/Xaccess:
#   localhost
# Then, this server is accessible at vncviewer <server>:0-5

service vnc-1024x768x16
{
        disable         = no
        type            = UNLISTED
        port            = 5900
        socket_type     = stream
        wait            = no
        user            = nobody
        group           = tty
        server          = /usr/bin/vncts
        server_args     = -geometry 1024x768 -depth 16
}

service vnc-800x600x16
{
        disable         = no
        type            = UNLISTED
        port            = 5901
        socket_type     = stream
        wait            = no
        user            = nobody
        group           = tty
        server          = /usr/bin/vncts
        server_args     = -geometry 800x600 -depth 16
}
```

(archivo truncado dada la extension del mismo)

Solo nos interesa la declaracion llamada 'service vnc-800x600x16 ', la cual es el servicio que estara escuchando en el puerto 5901. He elegido este puerto por compatibilidad con el resto de las resoluciones, ya que el puerto 5900 corresponde a la resolucion 1024*768 por ejemplo. Tambien podemos leer que se hace referencia a los nros. 0 al 5, en total 6 resoluciones de pantalla distintas:

```
1024*768*16    (0)
800*600*16     (1)
640*480*16     (2)
1024*768*8     (3)
800*600*8      (4)
640*480*8      (5)
```

No necesitamos tocar nada de aqui.
El argumento 'server_args' declara opciones pasadas al vuelo al servidor VNC una vez que se haya hecho efectiva su llamada por demanda mediante xinetd. Este argumento se completa en el archivo /etc/sysconfig/vncts el cual contiene la siguiente informacion:

```
# less /etc/sysconfig/vncts

## VNC Terminal Services Configuration
# Where is the Xvnc executable?
XVNC=/usr/bin/Xvnc

# Default options to use
VNCTSOPTIONS="-inetd -once -query localhost -SecurityTypes None"
```

Cada una de estas opciones, desde 'service vnc-800x600x16 ' hasta 'server_args' y 'VNCTSOPTIONS', paso a describirlas:

`service vnc-800×600x16`: Es el nombre del servicio.

`disable = no`: Tal como te puedes imaginar, podemos inhabilitar o no el servicio.

`type = UNLISTED`: Le indica a Xinetd que este servicio no se encuentra declarado en /etc/services.

`port = 5901`: puerto por el cual escucha este servicio

`socket_type = stream`: indica en que se basara el servicio, en este caso en transmision.

`wait = no`: Esta es la definicion principal para un ambiente de multiples peticiones, si declaramos aqui 'yes', obligamos al servicio que mantenga una conexion persistente y no la suelte hasta que el usuario haya terminado la sesion o haya muerto el servicio. En otras palabras, 'yes' significa 'solo una conexion a la vez'. Con 'no' habilitamos el modo de conexion 'multi-thread' o 'multi-hilo'. En otras palabras acepta varias conexiones a la vez.

`user = nobody`: Aqui declaramos que cualquiera puede acceder al servicio al no puntualizar ningun usuario en particular.

`group = tty`: ?

`server = /usr/bin/vncts`: Determina el ejecutable que lanzara este servicio.

`server_args`: Determina los argumentos que se pasaran al servicio una vez ejecutado, el cual se agrega a los por defecto declarados en /etc/sysconfig/vncts, paso a describirlas a todas las opciones:

•*`-inetd`*: determina que xinetd maneje a vncserver.

•*`-query localhost`*: indica que unicamente localhost esta habilitado para dar este servicio.

•*`-geometry 800x600`*: como puedes ver, indica el tamaño en pixeles horizontales por pixeles verticales que tendra la pantalla grafica una vez ejecutado el servicio.

•*`-depth 16`*: es la profundidad de colores, en este caso de 16 bits.

•*`-cc 3`*: clase de color. Existen dos variantes: -cc 2 y -cc3.(*)

•*`-once`*: solo una instancia de Xvnc sera ejecutada por este servicio.

•*`-securyitytypes`*: VNCSERVER no ejecutara la aplicacion 'vncpassword'.

•*`-fp unix/:7100`*: Servidor de fonts, por defecto.(*)

(*) ***NOTA: Como veras, estas opciones no aparecen, pero forman parte del antiguo standard de K12LTSP en las viejas versiones de Fedora Core y en la actual version de CentOS K12LTSP 5.3 EL, ¡ayudan tambien!. Tenlas en cuenta y experimenta con ellas.***

****NOTA: como puedes ver, hay un numero que se repite varias veces, el numero '16'. Este numero es la profundidad de color de SVGA, o sea 2^16 = 65.536 colores. Puedes bajar 1 bit la profundidad (15) y notaras que la performance de la renderizacion de la pantalla en el cliente, se vera muy levemente mejorada. Usala si necesitas realmente obtener esa minima performance extra. Esto sucede porque 2^15 = 32.768 colores, o sea que tanto vnc-server en el servidor, como svncviewer en la maquina cliente, tienen que traducir la mitad de colores con una profundidad de 15 bits con respecto a la profundidad de colores de 16 bits. Esto es util ya que recuerda que nos estamos manejando en una antigua red 10baseT y mas informacion por esta equivale mas lentitud al final de la cadena, ademas, si estas usando arquitectura i486SX (sin co-procesador matematico), es fundamental.****

Ahora, es momento de agregar a /etc/services el servicio llamado 'vnc-800x600x16':

```
# vim /etc/services
```

Aqui estamos explorando este largo archivo, vayamos al final del mismo y agreguemos:
(i)

```
vnc-800x600x16  5901/tcp
```

Listo. Podemos averiguar si hay mas puertos destinados a VNC:

```
(/)vnc
vnc-server      5900/tcp                # VNC Server
vnc-server      5900/udp               # VNC Server
```

(esc):wq
Ahora solo nos resta reiniciar xinetd:

```
# service xinetd restart
# init 3
```

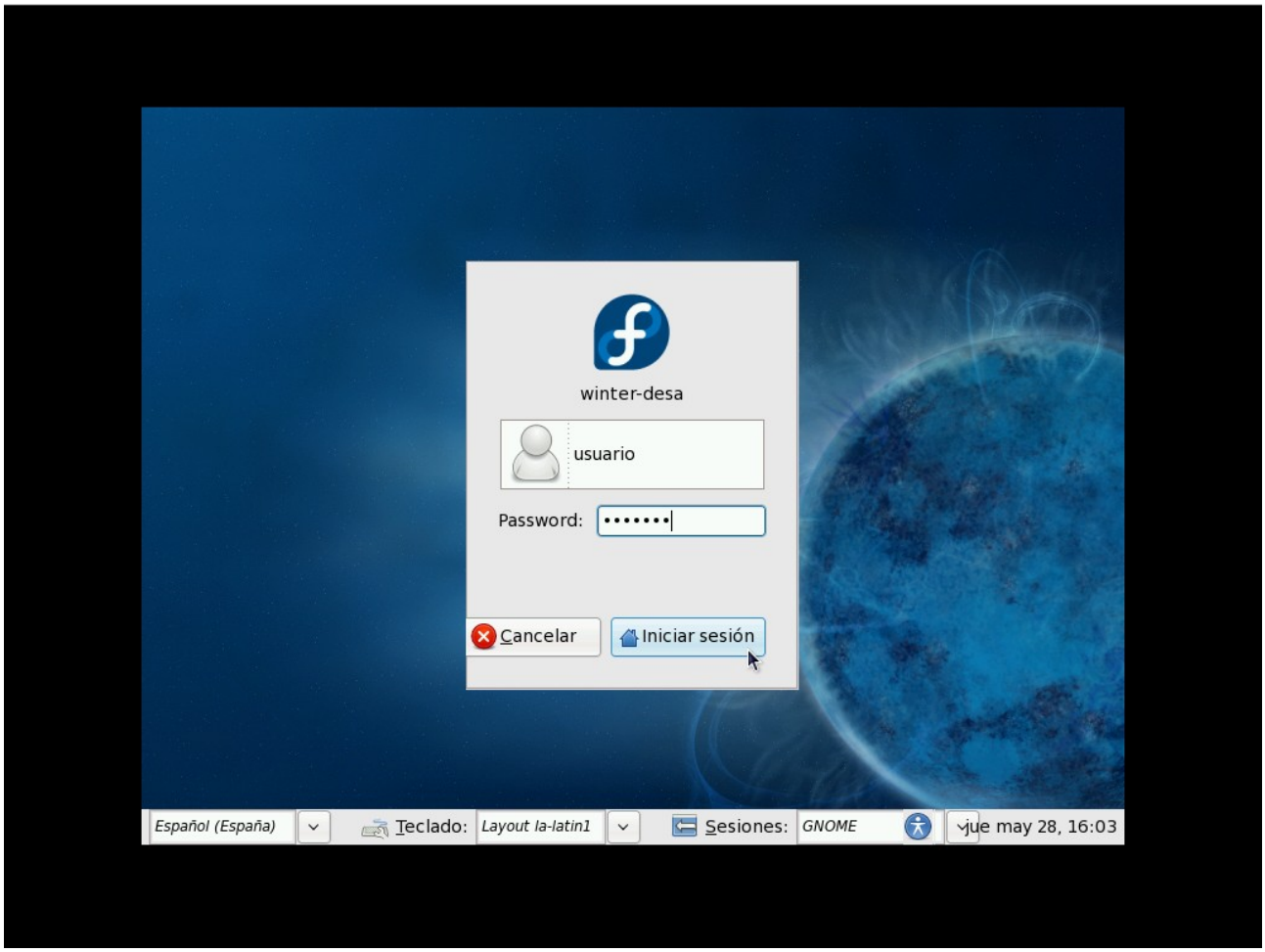
doy de baja el demonio gdm

```
# init 5
```

vuelve al entorno grafico , levanto gdm

¡Hemos terminado!

Con esto doy por concluida la primer parte acerca de la implementacion de un servidor VNC por demanda en Fedora 10, para ser usado con tu microdistribucion alojada en ese pequeño diskette. Aqui te dejo un pantallazo del ingreso via VNC:



¡Rueda VNC!

Supongo que estas prendiendo la maquina cliente, has insertado el diskette y una vez terminado el booteo has encontrado el grafico que estas viendo...

Implementando un Servidor UTC en CentOS 4.* & 5.*

Esta distribucion es muy estable y realmente es un logro importantisimo para la comunidad de Software Libre el hecho de contar con ella. Apoyate en la implementacion de Fedora 10 para abocarte a la implementacion de esta distribucion.

xinetd y vncserver

Lo primero que vamos a hacer es instalar vncserver, que seguramente podremos encontrarlo como 'vnc-server'. Si usas la linea GNU de la rama clones Red Hat, intenta con 'yum search vnc'.
En este ejemplo voy a usar la version 5.* de CentOS, la cual trae el paquete **vnc-server-4.1.2-9.el5**.

xinetd (eXtended InterNET Daemon), es un servicio que administra toda la conectividad basada en internet, escuchando puertos especificos de cada aplicacion que lo requiera, en este caso VNC. El servicio xinetd es una mejora del servicio inetd. Recomiendo visitar www.xinetd.org para mas detalle de este super-servicio.
En CentOS encontramos que el paquete xinetd es **xinetd-2.3.14-10.el5**.

Buscando VNC

Una vez que hemos instalado a vncserver en el sistema operativo, deberemos chequear si VNC esta presente como servicio. Esto lo vemos haciendo un:

```
# ls /etc/init.d
```

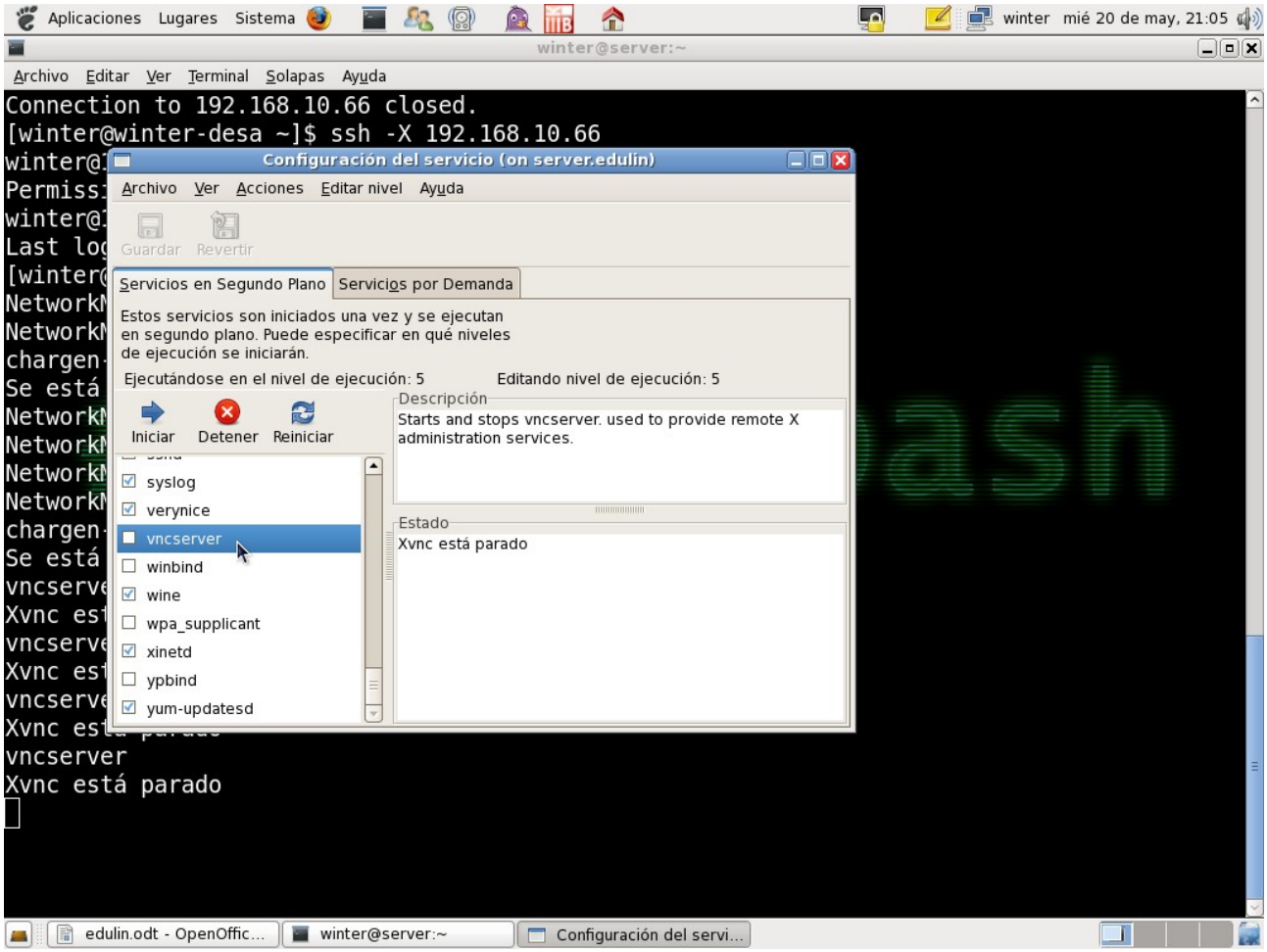
Buscamos el archivo 'vncserver', si lo encontramos, desde la gui grafica de GNOME en la opcion 'servicios' ('Accesorios > Configuracion de sistema > Configuracion de servidores > servicios' o 'Sistema > Administracion > Servicios') chequeamos si esta activo o sino, **debe estar desactivado**.

****NOTA: En realidad puede estar activado vncserver en nuestra grilla de servicios y de esta forma trabajara ademas, de por demanda por medio de xinetd, en modo solitario. Mi consejo: desactivalo.****

En consola para las distrubuciones tipo RH (Red Hat) podemos acceder de la siguiente forma:

```
# system-config-services
```

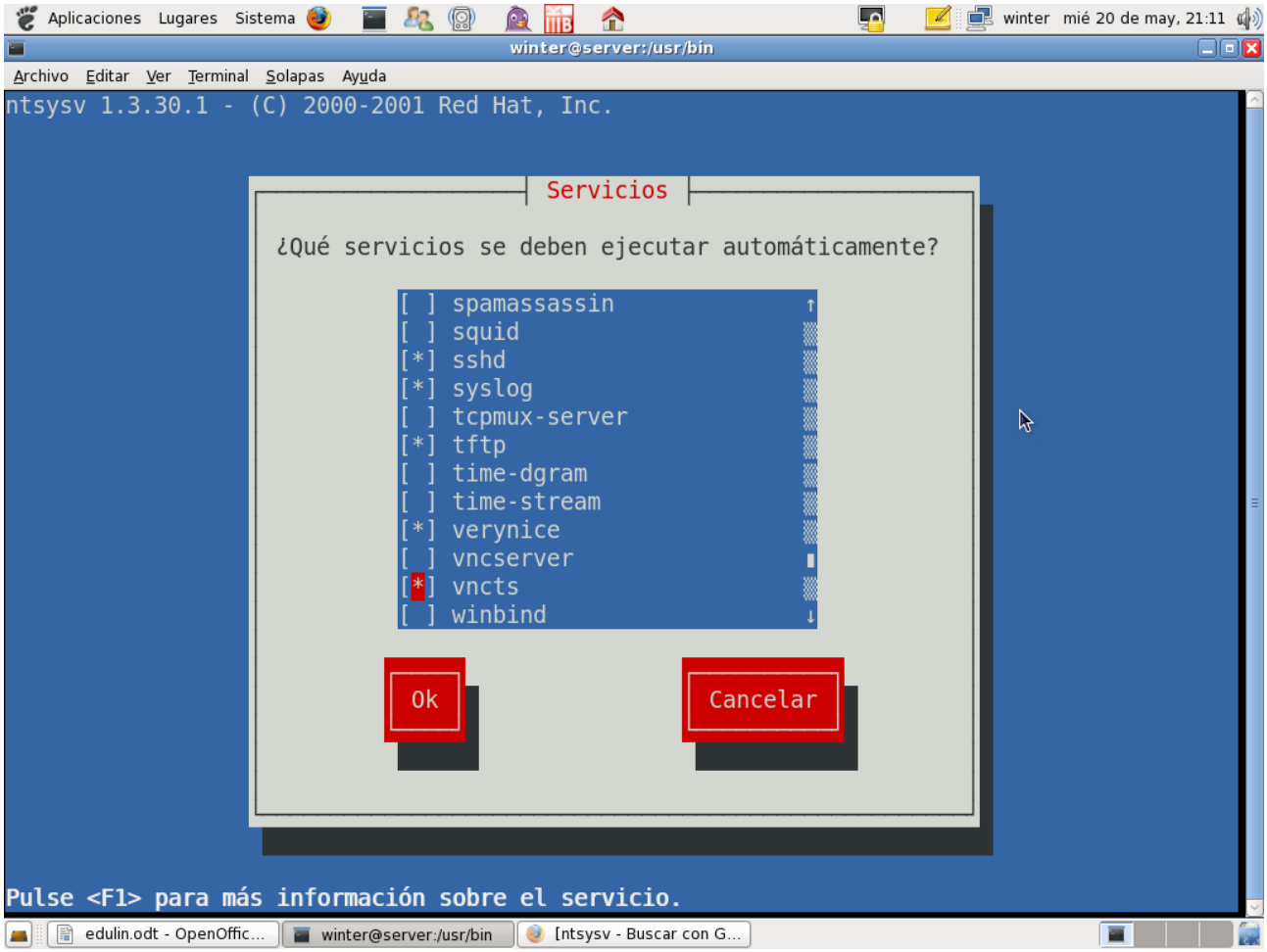
Nos devolvera, luego de unos instantes de chequear los servicios activos o no, esta pantalla:



Guardamos los cambios.

Otra forma de lograr lo mismo es usar el comando 'ntsysv', que basicamente es una herramienta diseñada para hacerle la vida mas facil al administrador del sistema. Se basa en ncurses:

```
# ntsysv
```



Ahora, es momento de configurar xinetd para que vnc quede en estado ‘ON DEMAND‘ , o sea que escuche al puerto correspondiente, pero solo entregue una conexion valida cuando alguien se la pida.
Para eso debemos configurar los servicios de nuestro sistema y el demonio que maneja a estos servicios (xinetd).

Configurando xinetd en CentOs 5.*

Abrimos una consola como root o entramos al modo grafico como tal.
Si usamos la consola vamos a usar un editor de texto como el ‘nano’ o 'vim' por ejemplo.
Si lo hacemos en modo grafico, podemos usar directamente; si usamos GNOME; gedit (Aplicaciones > Accesorios > Editor de textos).

/etc/services es el archivo de configuracion de todos los servicios que seran controlados por xinetd. Tiene la siguiente estructura:

Service-name *Nombre del servicio*	port/protocol *puerto/protocolo*	[aliases ...] *[alias...]*	[# comment] *[#comentario]*

lo abrimos y vamos al final del archivo y agregamos donde dice ‘# Local services’:

```
#VNC On-Demand
vnc-800×600x16          5901/tcp
```

Guardamos los cambios.
Como te daras cuenta, el nombre del nuevo servicio se llama ‘vnc-800×600x16′ y escucha el puerto 5901 con protocolo tcp.

Ahora, vamos a /etc/xinetd.d y creamos un archivo llamado vnc-800×600x16.

```
# touch vnc-800×600x16
```

Entramos a este archivo con el editor que querramos y agregamos lo siguiente:

```
service vnc-800×600x16
{
disable = no
protocol = tcp
socket_type = stream
wait = no
user = nobody
server = /usr/bin/Xvnc
server_args = -inetd -query localhost -geometry 800×600 -depth 16
-cc 3 -once securitytypes=none -fp unix/:7100
}
```

Voy a explicar que significa todo esto:

`service vnc-800×600x16`: Es el nombre del servicio.

`disable = no`: Tal como te puedes imaginar, podemos inhabilitar o no el servicio.

`protocol = tcp`: Este es el tipo de protocolo de red que usara por defecto este servicio.

`socket_type = stream`: indica en que se basara el servicio, en este caso en transmision.

`wait = no`: Esta es la definicion principal para un ambiente de multiples peticiones, si declaramos aqui 'yes', obligamos al servicio que mantenga una conexion persistente y no la suelte hasta que el usuario haya terminado la sesion o haya muerto el servicio. En otras palabras, 'yes' significa 'solo una conexion a la vez'. Con 'no' habilitamos el modo de conexion 'multi-thread' o 'multi-hilo'. En otras palabras acepta varias conexiones a la vez.

`user = nobody`: Aqui declaramos que cualquiera puede acceder al servicio al no puntualizar ningun usuario en particular.

`server = /usr/bin/Xvnc`: Determina el ejecutable que lanzara este servicio.

`server_args` = ***-inetd -query localhost -geometry 800×600 -depth 16 -cc 3 -once securitytypes=none -fp unix/:7100***: Determina los argumentos que se pasaran al servicio una vez ejecutado.

•-***inetd***: determina que xinetd maneje a vncserver.

•-***query localhost***: indica que unicamente localhost esta habilitado para dar este servicio.

•-***geometry 800x600***: como puedes ver, indica el tamaño en pixeles horizontales por pixeles verticales que tendra la pantalla grafica una vez ejecutado el servicio.

•-***depth 16***: es la profundidad de colores, en este caso de 16 bits.

•-***cc 3***: clase de color. Existen dos variantes: -cc2 y -cc3.

•-***once***: solo una instancia de Xvnc sera ejecutada por este servicio.

•-***securyitytypes***: VNCSERVER no ejecutara la aplicacion 'vncpassword'.

•-***fp unix/:7100***: Servidor de fonts, por defecto.

****NOTA: como puedes ver, hay un numero que se repite varias veces, el numero '16'. Este numero es la profundidad de color de SVGA, o sea 2^16 = 65.536 colores. Puedes bajar 1 bit la profundidad (15) y notaras que la performance de la renderizacion de la pantalla en el cliente, se vera muy levemente mejorada. Usala si necesitas realmente obtener esa minima performance extra. Esto sucede porque 2^15 = 32.768 colores, o sea que tanto vnc-server en el servidor, como svncviewer en la maquina cliente, tienen que traducir la mitad de colores con una profundidad de 15 bits con respecto a la profundidad de colores de 16 bits. Esto es util ya que recuerda que nos estamos manejando en una antigua red 10baseT y mas informacion por esta equivale mas lentitud al final de la cadena.****

Casi listo! solo nos queda habilitar XDMCP (Aplicaciones > configuracion de sistema > Pantalla de registro > XDMCP) y reiniciar los servicios afectados:

vamos a la consola y ejecutamos:

```
# service xinetd restart
```

Esto reinicia al demonio de Internet.

```
# init 3
```

Da de baja el demonio gdm.

```
# init 5
```

Vuelve al entorno grafico , levanto gdm

Con esto no necesitamos reiniciar la maquina.

Con esto doy por concluida la primer parte acerca de la implementacion de un servidor VNC por demanda en CentOS 4/5, para ser usado con tu microdistribucion alojada en ese pequeño diskette.

Aqui te dejo un pantallazo del ingreso via VNC:

¡Rueda VNC!

Supongo que estas prendiendo la maquina cliente, has insertado el diskette y una vez terminado el booteo has encontrado el grafico que estas viendo...

En el proximo apartado comienzo a explicar la segunda parte de la implementacion de un Servidor con los demas servicios, necesarios para la segunda implementacion de cliente VNC que no requiere un diskette con kernel+rfs y que me parece, es la mejor implementacion.

Comenzamos la instalacion con una nueva distribucion: OpenSuSe. Aclaro que practicamente es indistinta la distribucion a estas alturas de implementacion, asi que es aplicable a cualquiera, luego, volvere a explicar de una forma mas detallada la confuguracion de los servicios restantes, asi de esta forma, te fortaleceras en la implementacion en cualquier sistema GNU/LINUX.

Implementando un Servidor UTC en OpenSuse 10

Quiero creer que esta implementacion sirve para cualquier version futura de OpenSuse.

****NOTA: lo que sigue a continuacion esta basado en un documento que yo mismo escribi en el año 2007, para un cliente que me pidio implementar LTSP en OpenSuse10, cuando todavia no existia esa posibilidad de forma nativa en esa distribucion****

Arranque y configuracion

Voy a explicarte paso a paso, desde el comienzo, como implementar un Server OpenSuSe version 10:

•Encendemos la maquina.
•Arrancamos con el CD 1 elegimos “Instalacion”
•El CD ejecuta sus aplicaciones de diagonostico y comienza la instalacion:
>Idioma: elegimos : “ESPAÑOL”. **Clic en “SIGUIENTE”**.
>Comprobacion de medios: comprueba el estado de los CDs es una herramienta util, pero no creo que lo sea para este caso, ya que aceptamos que los isos que grabamos se encuentran en buenas condiciones. **Clic en “SIGUIENTE**”.
•Acuerdo de licencia: la licencia expresa de NOVELL. Open Source. **Clic en “SIGUIENTE**”.
•Modo de instalacion: elejimos “NUEVA INSTALACION”. **Clic en “SIGUIENTE**”.
•Reloj y zona horaria: en “ZONA” elegimos “America cenntral”, en Zona horaria “Buenos Aires”.
•Cambiamos fecha y hora si es necesario haciendo clic en el boton “Cambiar” de “FECHA & HORA”.
•Cambiamos a “HORA LOCAL” en “RELOJ DE HARDWARE…”. Clic en “SIGUIENTE”.
•Seleccion de escritorio: tenemos 3 opciones: KDE, GNOME u OTRO. Personalmente de entrada me inclino a GNOME que es un GRAN entorno grafico e intuitivo, pero mas tarde probaremos el “SISTEMA GRAFICO MINIMO” que ofrece la opcion “OTRO”. elegimos GNOME. Clic en “SIGUIENTE”.

Configuracion de la instalacion

probaremos instalar desde cero un unico sistema operativo con sistyema de archivos REISERFS que es por defecto lo que suse 10 ofrece. A partir de ahora comienza la instalacion.
Espacio requerido: 2.4 GB de disco.

•Una vez terminada la instalacion, llenamos los campos restantes: contraseña de root y su cifrado lo deje tipo DES.
•Configuracion de RED. En la opcion cortafuegos: desabilita el cortafuegos. Esto es importante, ya que el CORTAFUEGOS (FIREWALL) puede denegar puertos que son necesarios para ejecutar ciertos servicios como DHCP o TFTP.
•En tarjeta de red cambiamos la configuracion DHCP por estatica con la IP 192.168.0.254 (para nuestro server).
•Cambiamos los nombres en nombre de host por: “server” y en nombre de dominio por algo que se nos ocurra, por ejemplo “utc”. Esto se hace porque la instalacion por defecto de LTSP utiliza el nombre “server” para el servidor y el dominio “utc” por defecto para su entorno.
•Si tenemos DNS de algun ISP lo ponemos en “servidor de nombres nn”.
•En “enrutado” si tenemos alguna pasarela (gateway) la colocamos aqui.

Todo lo demas que vemos en “configuracion de red” lo dejamos por defecto como esta.
Ahora se guardara la configuracion de la red y se reiniciaran algunos servicios relacionados con la conectividad del equipo:

•La opcion 'probar conexion a Internet'. lo pasamos por alto.
•Metodo de autentificacion de usuarios. Lo dejamos por defecto en “local”.
•Ahora el instalador nos pide que ingresemos un “nuevo usuario local”, lo cual lo haremos para probar nuestro servidor mas adelante. Destildamos el “inicio de sesion automatico”.
Notas de la version y configuracion de hardware:
Aqui podemos leer lo ultimo que trae opensuse. Con respecto a la deteccion de hardware, fue sin problemas. Ha detectado la placa de red, de video y demas sin ningun problema.

Inicio de sesión

Entramos a la pantalla grafica de login como root. ¡OJO a no asustarse!, si sigues paso a paso este documento no tienes de que preocuparte, simplemente usamos el entorno grafico como 'root' para poder editar de forma rapida con gedit, ya que acepta los comandos Ctrl-c; Ctrl-v y Ctrl-x. De todas formas, si no te sientes comodo o seguro, ingresa como el usuario que declaraste en la instalacion, abre un terminal en modo grafico y luego utiliza el comando 'su -', usa gedit ahora como superusuario.

Instalando los servicios necesarios

Lo primero que vamos a hacer es instalar `nfs-server, tftp, vnc-server y dhcp-server`. Esto deberemos hacerlo como siempre desde Yast, la herramienta de gestion de OpenSuSe, xinetd ya viene instalado por defecto.

NOTA:***Una forma tambien seria el elegir todos estos paquetes a la hora de la instalacion, eso lo dejo a tu criterio, lo que te parezca mas comodo o mas divertido.***

Configurando nuestros servicios

DHCP-SERVER

Lo arrancamos, paramos y rearrancamos con:

```
# /etc/init.d/dhcpd [start-stop-restart]
```

Apenas instalado nos dara el error siguiente:

```
“DHCPD_INTERFACE en /etc/sysconfig/dhcpd esta vacia!”
```

Este archivo de configuracion es analogo al de Debian que es /etc/defaults.
Entramos al archivo dhcpd y donde encontramos la variable DHCPD_INTERFACE, agregamos entre las comillas “eth0″, guardamos los cambios y rearrancamos dhcp.

Ahora el error que nos dara es que no encuentra la red actual, eso es logico porque el archivo dhcpd.conf por defecto apunta a otra red.
Solo tenemos que renombrar el archivo de configuracion que se encuentra en /etc/dhcpd.conf por dhcpd.conf.old y crear un nuevo dhcpd.conf:

```
# cd /etc
# mv dhcpd.conf dhcpd.conf.old
# touch dhcpd.conf
```

Abrimos gedit:

```
# gedit dhcpd.conf
```

Como podemos ver esta vacio el nuevo archivo de texto, insertamos lo siguiente:

```
##############################################
###   ARCHIVO DE CONFIGURACION DE DHCP    ###
##############################################
default-lease-time 21600;
max-lease-time 21600;
ddns-update-style none;
allow booting;
allow bootp;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers  192.168.0.254;
option domain-name-servers 192.168.0.254;
next-server 192.168.0.254;
option domain-name  "utc";
option root-path  "/opt/utc";
option option-128 code 128 = string;
option option-129 code 129 = text;
shared-network WORKSTATIONS {
     subnet 192.168.0.0 netmask 255.255.255.0 {
          range dynamic-bootp 192.168.0.200 192.168.0.253;
          use-host-decl-names on;
          option log-servers 192.168.0.254;
          }
     }
group     {
use-host-decl-names on;
option log-servers 192.168.0.254;
}
```

Reiniciamos dhcpd y comprobamos que no nos de ningun, error, si obtenemos alguno, nos debemos fijar en alguna omision en el ingreso del codigo.

Breve explicacion del archivo de configuracion dhcpd.conf

Parametros exclusivos del host:

`default-lease-time`: especifica la cantidad de tiempo en segunos que el cliente mantendra la IP otorgada por dhcp.

`max-lease-time`: igual que el anterior, solo que aqui indicamos el tiempo maximo permitido.

`ddns-update-style:` indica a dhcp si tiene que actualizar el servidor DNS cuando se realiza un prestamo o liberacion de IP al cliente.

`allow booting`: indica a dhcp que debe permitir peticiones tipo multicast (broadcasting) de nodos pidiendo identificacion de red.

`allow bootp`: indica a dhcp que permita peticiones de identidad del tipo bootp (etherboot).

`next-server`: indica a dhcp la existencia de otro servidor dhcp, en este caso al ser una red simple, tipo clase C, el valor de la direccion IP es la del mismo host.

`shared-network WORKSTATIONS {}`: comienza aqui la declaracion de red compartida llamada WORKSTATIONS, la cual configura con 'subnet {}' el numero de la red, que siempre termina en 0 (cero), asi en este ejemplo tenemos declarada a subnet como 192.168.0.0. Configuramos tambien la mascara de la red con 'netmask', la cual sera siempre para estos casos 255.255.255.0.

Ahora vemos que se abre una nueva declaracion: range dynamic-bootp.

`range dynamic-bootp`: declara el rango de direcciones IP que podran ser entregadas a los nodos que asi las soliciten, en este caso se declaro que el rango va desde la direccion IP 192.168.0.200 a la direccion IP 192.168.0.253.

`use-host-decl-names`: indica a dhcpd que el nombre provisto por la declaracion de host sera el hostname de la maquina cliente, por ejemplo:

```
group {
                use-host-decl-names on;
                host joe {
                  hardware ethernet 08:00:2b:4c:29:32;
                  fixed-address joe.fugue.com;
                }
              }
Lo anterior es equivalente a:
                host joe {
                  hardware ethernet 08:00:2b:4c:29:32;
                  fixed-address joe.fugue.com;
                  option host-name "joe";
                }
```

`group {}`: es utilizado para aplicar uno o mas parametros a un grupo de declaraciones. Esta es la declaracion de grupo de caracteristicas similares entre hosts, tambien podemos agrupar hosts a los cuales declaramos parametros similares.

Parametros exclusivos para los clientes

Estos parametros se usan para asignar a las maquinas clientes caracteristicas especiales con la palabra clave 'option' seguida de un opcion de nombre y una opcion de dato. Estas declaraciones seguidas de 'option' inciden directamente sobre el host cliente.

***`option subnet-mask [IP]`*:** especifica la mascara de red que se aplicara al cliente.
***`option broadcast-address[IP]`*:** especifica la direccin de broadcast que se aplicara al cliente.
***`option routers[IP]`*:** especifica los gateways al que podra acceder el cliente.
***`option domain-name-servers[IP]`*:** especifica una lista de DNS disponibles para el cliente.
***`option domain-name “texto”`*:** especifica el nombre de dominio que usara el cliente cuando necesite resolver nombres a través de DNS.
***`option root-path  "texto"`*:** especifica el path donde el cliente debe buscar su Root file system.

option option-128 code 128 = string: se utiliza para enviar parametros al kernel
option option-129 code 129 = text: se utiliza para enviar parametros al kernel
option log-servers[IP]: especifica una lista de servidores de logs UDP disponibles para el cliente.

Terminada la explicacion de nuestro dhcpd.conf, continuamos.

Reiniciamos nuevamente dhcpd. Debemos tener mensaje de “done”.

TFTP Server

Basicamente, reiniciando el demonio xinetd es suficiente, de todas formas debemos saber que tftp debe apuntar al directorio /tftpboot que es el que creó la instalacion de LTSP y es donde se alojan los kernels de arranque del entorno en el cliente.
En /etc/xinetd/tftp debemos comprobar que este apuntando el servicio a /tftpboot corriendo en modo seguro (-s). El modo seguro le dice a tftpd que el path a donde apuntan los ficheros es ABSOLUTO, no relativo.

NFS Server

Vamos a /etc/exports y cambiamos /opt/ltsp por /opt/utc (podemos darnos cuenta que es la misma linea de dhcpd.conf en option-root-path).

Reiniciamos el servicio:

```
# /etc/init.d/nfsserver restart
```

Y comprobamos los directorios compartidos:

```
# showmount -e
```

Veremos listados los directorios de /etc/exports

Reiniciamos Xinetd:

```
# /etc/init.d/xinetd restart
```

Configuracion de XDMCP

Vamos a Aplicaciones > Sistema > Configuracion > Configuracion de pantalla de entrada (Login Screen Setup)

Vamos a la pestaña “general” y habilitamos un tema para la entrada remota, vemos que previamente lo teniamos desabilitado.

Vamos a “seguridad” y comprobamos que XDMCP este habilitado.

En la pestaña “XDMCP” le decimos cuantos clientes vamos a colgar a gdm en “Maximo de sesiones remotas”.

Por último, vamos a Yast y desinstalamos el generador de logs del sistema que es “syslog-ng”.
Una vez desinstalado, instalamos “syslog” (se encuentra en el CD 1).
Una vez instalado vamos a /etc/sysconfig/syslog y en la linea 20 aprox. encontramos una declaracion de variable:

```
SYSLOGD_PARAMS=”"
```

Lo reemplazamos por:

```
SYSLOGD_PARAMS=”-r”
```

Esto es para que sea posible la generacion remota de logs. Syslogd de esta forma escucha a la red. Por defecto syslog esta en modo “standalone” o sea local.

REINICIAMOS EL SERVER.
Fin de la implementacion en Open SuSe 10.0

****NOTA: es de imperiosa necesidad que si usas una distribucion como Debian, Ubuntu o SuSe, desinstales inetd y lo reemplaces por xinetd, ya que por experiencia propia te puedo advertir que en el banco de pruebas inetd funciona como debe, pero en la practica recibiendo 10 peticiones al mismo tiempo de cara al servidor VNC, suele suceder a veces, que las conexiones a los clientes colapsan.****

CAPITULO 5: Prescindiendo del diskette para UTC

Eliminando el Diskette o “como transportar Kernel & RFS por red”

Como se explico anteriormente, el Servidor necesita tener habilitados servicios de transporte para poder ejecutar exitosamente la tarea de conexion cliente/servidor.

El problema que se me presento, una vez desarrollado este microsistema, fue el de manutencion, o sea, la idea siempre fue minimizar la tarea de mantenimiento ya sea del Servidor, como de las maquinas clientes.

Lamentablemente, el “talon de Aquiles” de esta implementacion siempre fue el uso del diskette, ya que aparte de ser de muy mala calidad actualmente, estando al borde de la desaparicion, su tiempo de vida util es tremendamente corto, a comparacion de los viejos diskettes.

La idea aqui entonces es poder aprovechar disketts en mal estado, pero que contengan sus primeros 100kb optimos, una ROM ethernet insertada en la placa de Red ISA o un disco duro ya obsoleto.

Yo pretendo mostrar la primera y tercera opciones (la tercera opcion, dada su complejidad la explico en el ANEXO 7), dada la simplicidad de construccion e implementacion en el caso #1 y principalmente por costos economicos en el caso #3, ya que la segunda opcion es sustancialmente elevada en precio, ya que necesitamos conseguir memorias ROMs compatibles con la velocidad del bus de datos y direcciones de estas viejas placas ISA y un grabador de EPROMS para poder insertar el codigo ejecutable de la rutina de arranque compatible con la placa de red ISA ne2000. El precio de cada ROM puede llegar a costar hasta U$s30 por cada una y un grabador de EPROM puede llegar a costar U$s250. Realmente entre ROMs y grabador tenemos el valor equivalente a cerca de 20 clientes i486, lo cual sinceramente no se justifica de ninguna manera este gasto, asi que optamos por las otras opciones.

NOTA:*** En una futura revision, junto con la opcion #3 habra que incluir una forma de apagar el disco duro, una vez que 'init' arranque, ya que no esta respetando la filosofia de equipamiento silencioso y ecologico con un disco constantemente girando y consumiendo energia electrica, aunque solo se haya usado en los primeros segundos del arranque del cliente y nada mas. La herramienta que quizas sea la adecuada podria ser 'hdparm'***

Antes que nada, debemos preparar nuestro kernel, o sea, parcharlo para que pueda ser transportable por la red, cuando la rutina de arranque inicial asi lo requiera y ademas preparar nuestro DHCP server para poder reconocer la placa de red del cliente, poder reconfigurar el kernel al vuelo para que pueda obtener una direccion IP y asi de esta forma via NFS cliente/servidor, obtener el RFS completo, para luego arrancar init.

Preparando nuestro Kernel

Para que nuestro Kernel este optimizado para esta nueva funcionalidad, debe tener habilitadas ciertas funcionalidades de uso exclusivo para esta nueva metodologia, las cuales son:

RamDisk support
Soporte para disco RAM en memoria como anteriormente explique
NFS protocol support
Soporte para el protocolo de transporte de archivos nativo de linux NFS
root file system over NFS support
Soporte para el transporte del RFS via red por medio del prtocolo NFS
Kernel level IP autoconfiguration via BOOTP y DHCP

Autoconfiguracion de direccion IP para el Kernel mediante BOOTP y DHCP, esto se usa una vez que el kernel transportado fue desempaquetado en la memoria de la maquina cliente.

Estos parametros y opciones deberas incluirlos de la misma manera que lo hiciste cuando compilaste el Kernel al principio de este libro, vuelve a repasar el capitulo si tienes dudas.

Recomiendo en esta tarea para hacerla mucho mas llevadera instalando CentOs 5EL-k12ltsp de www.k12ltsp.org

Lo primero que vamos a hacer es crear un directorio dentro de /tftpboot, donde por defecto CentOs 5EL-k12ltsp aloja los kernels LTSP, este directorio puede tener cualquier nombre, yo lo llame "kernel", obvio que estamos frente a nuestro servidor flamante.

```
# mkdir kernel
```

Ahora vamos a copiar nuestra imagen llamada "bzImage", la cual quedo en nuestra maquina de desarrollo, podemos transportarla via scp (security copy) a nuestro servidor, en el nuevo directorio llamado "kernel".

Convirtendo a nuestro Kernel en una imagen booteable

Existe una herramienta a tal proposito llamada *mknbi*, desarrollada por la misma gente que desarrollo el proyecto ETHERBOOT, el cual es una serie de rutinas de arranque de placas de red, las cuales permiten cargar una imagen booteable en memoria.
Lo que hace especificamente *mknbi* es convertir al kernel en un kernel boooteable por red, lo cual es imprescindible para esta nueva variante.

> NOTA:***Deberas instalar *mknbi* si no esta en tu sistema, fijate si en los repositorios oficiales lo encuentras o utiliza www.google.com para encontralo en formato empaquetado para tu distribucion, o las fuentes y compilalo si es necesario, siguiendo las instrucciones del paquete.***

Bien, una vez que estamos dentro del directorio /tftpboot/kernel con el dentro, usamos la herramienta mknbi para parcharlo:

```
# chmod 0555 bzImage
# chown root:root bzImage
# mknbi-linux bzimage –output=kernel –rootdir=/opt/utc
```

en primera instancia hacemos ejecutable nuestra imagen, en segundo lugar hacemos propietario a root de nuestra imagen y por ultimo la parchamos, haciendola booteable con mknbi-linux, dandole un nombre a la nueva imagen resultante llamada 'kernel' e indicandole a esta nueva imagen que debera buscar su RFS en /opt/utc.

Ahora creamos en /opt un directorio llamado 'utc' el cual alojara nuestro RFS descomprimido, tal cual como lo creamos en la primer etapa al principio de este libro:

```
# mkdir /opt/utc
```

ahora copiamos nuestro RFS de la misma forma que lo hicimos con el kernel (con scp).
Para mas informacion: man scp

NOTA:*** ¡CUIDADO! USANDO *scp* SIN PARAMETROS, ROMPERA ABSOLUTAMENTE TODOS LOS ENLACES BLANDOS ENTRE LIBRERIAS EN NUESTRO RFS, INVESTIGA LAS OPCIONES -a Y -p QUE PRESERVAN EL ESTADO DE LOS ARCHIVOS.***

Listo! Nuestro kernel esta parchado y nuestro RFS esta instalado! Es hora de ajustar nuestro RFS a las nuevas necesidades y por ultimo configurar los servicios pertinentes, tales como NFS, TFTP y DHCP.

Dentro de nuestro directorio /opt/utc, nos dirigimos a etc/init.d y editamos rcS, quitamos la linea perteneciente a la declaracion de direccion IP que habiamos preparado con anterioridad para nuestro diskette VNC cliente. Ya que la anterior forma supone que debemos declarar por cada cliente una direccion IP fija y de esta forma crear tantas imagenes de RFS como clientes tengamos. Esto es engorroso, pero fue la unica forma en su momento para poder hacer funcionar 10 clientes en red con entorno grafico. En esta nueva variante no necesitamos nunca mas declarar la direccion IP en nuestro RFS, ya que este ahora es un unico y generico recurso el cual accederan todos los clientes, los cuales seran definidas sus direcciones IP mediante el servicio DHCP que mas adelante describire.

Habilitando el resto de los servicios

Servicio TFTP

Este servicio se utiliza para trasnportar el kernel desde el Servidor hacia la maquina cliente, inmediatamente despues de que la rutina de arranque etherboot pide via dhcp sus datos, como anteriormente en el libro explique. Solo se utiliza una vez.
por defecto, CentOs 5EL-k12ltsp trae configurado tftpd server para el uso de LTSP, vamos a aprovechar esto, solo inspeccionando que este todo en orden:

vamos a /etc/xinetd.d

```
# cd /etc/xinetd.d
```

```
# less tftp
```

```
service tftp
{
        disable = no
        socket_type = dgram
        protocol = udp
        wait = yes
        user = root
        server = /usr/sbin/in.tftpd
        server_args = -s /tftpboot
        per_source = 11
        cps = 100 2
        flags = Ipv4
}
```

Deberiamos encontrar esto declarado, aqui lo mas importante es el path declarado con 'server_args' donde se alojaran las imagenes del Kernel. Te recuerdo que con aterioridad explique que significa tftp. Una vez que el kernel se aloja en memoria del cliente, este se desempaqueta y se autoejecuta,

el segundo paso es transportar via NFS todo el RFS.
Para mas informacion: man tftpd.

Servicio NFS

Como dije antes, NFS es el protocolo de transporte de paquetes nativo de linux, el cual en este caso es usado por el cliente para obtener todo el sistema de archivos, nuestro RFS. Una vez ejecutado el cliente NFS que esta embebido dentro de nuestro Kernel (cuando lo configuramos en la opcion NFS > NFS root file system), éste, mediante el parche mknbi, sabe que tiene que pedirle al servidor el contenido del directorio que esta en /opt/utc.

Lo unico que debemos hacer, es declarar en nuestro servidor el path completo de acceso al nuevo recurso NFS:

```
# cd /etc
```

```
#vi exports
```

```
## LTS-begin ##
#
# The lines between the 'LTS-begin' and the 'LTS-end' were added
# on: Wed Nov 28 11:19:42 PST 2001 by the ltsp installation
script.
# For more information, visit the ltsp homepage
# at http://www.ltsp.org
#
/opt/ltsp/i386
192.168.0.0/255.255.255.0(ro,no_root_squash,sync)
/var/opt/ltsp/swapfiles
192.168.0.0/255.255.255.0(rw,no_root_squash,async)
```

Solo debemos agregar esta linea:

```
/opt/utc  192.168.0.0/255.255.255.0(ro,no_root_squash,async)
```

reiniciamos nuestro servicio:

```
# service nfs restart
```

Servicio DHCP

Como tambien explique, DHCP, es el servicio que se encarga de darles identificacion a cada maquina (direccion IP, gateway, DNS, etc), pero en este caso lo vamos a usar para, ademas de declarar una IP fija por cada maquina, enviar paramatros que seran interpretados al vuelo por el Kernel para autoconfigurarse, una vez desempaquetado y autoejecutado, ya que nuestro Kernel, cuando arranca tiene todas sus banderas a cero y por lo tanto no sabe que IP tendra ni donde encontrara el sistema de archivos, mas alla que eso haya sido declarado por mknbi.

Antes de configurar nuestro servicio DHCP, debemos tener en la cabeza la idea de declarar una IP fija por cada maquina cliente que colguemos a la red, para ello, deberiamos tener lista una maquina cliente con su placa de red ISA del tipo NE2000 por un lado, por el otro vamos a crear un diskette de arranque etherboot con la imagen universal que adjunto en las fuentes.
Esta imagen posee mas de 30 rutinas de arranque de conocidas placas de red ISA y PCI.
Copiamos donde querramos nuestra imagen llamada ebnet522.dsk, la cual esta basada en la version 5.2.2 de etherboot, luego, insertamos un diskett en blanco en nuestra maquina host, o donde tengamos esta imagen universal y tipeamos:

```
# cp ebnet522.dsk /dev/fd0
```

Con esto, copiamos en 'crudo' y sin montar, toda la imagen al diskette.

Ahora procedemos a abrir una consola en nuestro Servidor y escribimos:

```
# tail -f /var/log/messages
```

Este comando nos muestra en tiempo real por la salida standard (la pantalla) los mensajes de sistema. Lo que buscamos aqui es la famosa 'MAC' o identificador único que posee cada placa de red de fábrica. Este numero identificatorio es hexadecimal y esta dividido por ':' en serie de 6 pares:

```
00:D0:09:30:6A:1C
```

Esto debemos encontrar, unos segundos despues de encender nuestra maquina cliente, con el diskette universal insertado en su diskettera, no esta misma serie de 6 pares que puse como ejemplo, sino un identificatorio parecido. Una vez identificada la MAC del cliente, procedemos a anotar en papel o copiar por otros medios, dejo a la libre imaginacion la accion de copiar digitalmente esos numeros.

Una vez que hemos obtenido nuestra MAC vamos a configurar nuestro servicio DHCP, el cual en CentOs 5EL-k12ltsp se encuentra en /etc/dhcpd-k12ltsp.conf.

```
# vi /etc/dhcpd-k12ltsp.conf
# Sample configuration file for ISCD dhcpd
#
# Don't forget to set run_dhcpd=1 in /etc/init.d/dhcpd
# once you adjusted this file and copied it to /etc/dhcpd.conf.
#
 default-lease-time            21600;
max-lease-time                21600;
ddns-update-style none;
allow booting;
allow bootp;
option subnet-mask            255.255.255.0;
option broadcast-address      192.168.0.255;
option routers                192.168.0.254;
option domain-name-servers    192.168.0.254;
next-server                   192.168.0.254;
option domain-name            "";
option root-path              "192.168.0.254:/opt/utc";
option option-128 code 128 = string;
option option-129 code 129 = text;
option option-221 code 221 = text;
     shared-network WORKSTATIONS {
          subnet 192.168.0.0 netmask 255.255.255.0 {
               range dynamic-bootp 192.168.0.200 192.168.0.253;
               use-host-decl-names       on;
               option log-servers        192.168.0.254;
```

..... *texto truncado debido al tamaño* .....

Esto es lo mas importante, que debe estar declarado en el archivo de configuracion, especialmente las opciones 128 y 129, las cuales se utilizan para poder enviar parametros al vuelo al Kernel. Ahora, al final de nuestro archivo de configuracion y controlando no duplicar el nombre de los hosts clientes declarados agregamos:

```
host ws101 {
          hardware ethernet       00:50:fc:06:30:19;
          fixed-address           192.168.0.101;
          filename                "/kernel/kernel";
          option option-128 e4:45:74:68:00:00;
          option option-129 "ip=dhcp nfsroot=192.168.0.254:/opt/utc";
          }
```

El verdadero truco esta en la ultima declaracion , pero antes explico que significa todo:

hardware ethernet 00:50:fc:06:30:19
Esta es obviamente, la MAC de la placa de red

fixed-address 192.168.0.101
Esta es la direccion IP que se le asignara a nuestra maquina cliente mediante las preguntas que le efectua la rutina de arranque etherboot al servidor dhcp

filename "/kernel/kernel"
Esto le indica al demonio tftp donde tiene que buscar la imagen de nuestro kernel una vez que la rutina de arranque haya sido liberada.

option option-128 e4:45:74:68:00:00
Este es otro 'numero magico' que le indica al kernel que se prepare para recibir parametros a partir de este momento.

option option-129 "ip=dhcp nfsroot=192.168.0.254:/opt/utc
Estos son los parametros con truco que le indican al kernel que:

Pida nuevamente por dhcp direccion IP y todo lo que pueda darle de identificacion el servidor dhcp con el argumento 'ip' y el path completo a donde se encuentra el root file system con el argumento 'nfsroot'.

Reiniciamos nuestro servidor dhcp:

```
# service dhcpd restart
```

¡Con esto terminamos la implementacion de un cliente fino del tipo i486 con solo 4MB de memoria RAM! Solo queda encender la maquina cliente y esperar la pantalla de login grafica despues de aproximadamente 30 segundos.

Palabras finales

¡Gracias por llegar hasta aqui!
Ahora sabes con mas acierto y profundidad como funciona un sistema GNU/LINUX, porque has creado uno, desde sus fuentes, ahora nada te detiene para evolucionar y crear nuevos sistemas para nuevas necesidades... Estas en la antesala de los sistemas embebidos...

Alberto Castillo

ANEXOS

ANEXO 1 ¿Como funciona Etherboot?

Basicamente Etherboot es una rutina de arranque por red. Que permite a cualquier computadora con placa de red poder cargar un sistema operativo “al vuelo” totalmente en memoria.
El resultado final se divide en tres pasos:

- La computadora obtiene una identidad (direccion IP)
- La computadora obtiene un sistema operativo (Kernel)
- La computadora obtiene un sistema de archivos para operar (Root File System)

Para el primer paso, el servidor donde se aloja tanto el Kernel como el Root File System, deben saber identificar a la maquina en la cual enviaran los paquetes por la red, para esto, la misma placa de red posee un identificativo unico e intransferible, una direccion de 48 bytes en seis pares hexadecimales separados por ':', por ejemplo:

`00:60:08:C7:A3:D8`

A esto se le denomina 'MAC ADDRESS'.
Como sabras, es necesario un protocolo standard de comunicacion para poder lograr con exito el envio en primera instancia, de los paquetes correspondientes para la identificacion del cliente, o sea, una direccion IP. El protocolo se llama BOOTP (Boot Protocol).
La evolucion de BOOTP es por ende DHCP, pero en este caso (EtherBoot), el protocolo principal es BOOTP.

Todo comienza de esta manera:

- Se enciende la maquina cliente y se comprueba mediante la BIOS el estado general del equipo.
- La rutina de arranque Etherboot, se autoejecuta. Lo que reside dentro de esta rutina es un cliente bootp/dhcp y el driver correspondiente a la placa de red de turno.
- Una vez Etherboot se ejecuta permite que el minicliente bootp/dhcp tome el control de la situacion.
- El minicliente bootp/dhcp toma nota de la MAC ADDRESS.
- Luego se genera una señal de 'Broadcasting' o Multidifusion por red, o sea: se envia una peticion general de busqueda, para encontrar un servidor en la misma red, con esta peticion “Hola! Soy la direccion MAC `uu:vv:ww:xx:yy:zz`, por favor enviame mi direccion IP correspondiente”.
- El servidor que posee el servicio BOOTP/DHCP, esta constantemente escuchando por alguna peticion. La recibe y comprueba en una lista propia si tiene declarada la direccion MAC de la maquina cliente.
- Si existe en la lista del Servidor la MAC ADDRESS, automaticamente, se le envia todos los datos identificatorios correspondientes “Tu nombre de red es ws001, tu direccion IP es 192.168.0.1, tu Gateway 192.168.0.254 la imagen de sistema operativo que tenes que bajarte es /tftpboot/kernel”.
- Ahora es momento de traer la imagen del Kernel (o sistema operativo) al cliente, para esto debemos usar otro protocolo de red llamado TFTP (Trivial File Transfer Protocol). Basicamente, TFTP es una version reducida de FTP, no contempla ningun tipo de autentificacion y trabaja en modo UDP (User Datagrama Protocol)en vez de TCP (Transmision Control Protocol). Se elige usar UDP en vez de TCP por la simplicidad de uso,

al no ser necesarias ninguna autenticacion de los datagramas que viajan hacia la maquina cliente, ya que estamos transmitiendo datos a nivel hardware, no a nivel usuario. Ademas UDP es mucho menor en tamaño que TCP y ayuda de esta forma a poder entrar dentro de los tipicos 16KB de tamaño de las rutinas de arranque Etherboot.

- UDP es un protocolo orientado a la transmision por bloques, no en cadena como TCP, asi que basicamente, transportamos bloque a bloque todo nuestro kernel a la maquina cliente:

```
UDP: “Necesito bloque Nro.1 de /tftpboot/kernel”
SERVER: “Enviando bloque Nro.1 de /tftpboot/kernel”
UDP: “Necesito bloque Nro.2 de /tftpboot/kernel”
SERVER: “Enviando bloque Nro.2 de /tftpboot/kernel”
UDP: “Necesito bloque Nro.3...”
```

  Si falla la transmision Rx/Tx de algun bloque, este se envia nuevamente despues de un tiempo preestablecido. Una vez treminada la descarga de todos los bloques, el Kernel toma el control de la situacion.

- Se descomprime el Kernel y se autoejecuta, comprobando el buen funcionamiento de todos los dispositivos que posee declarados en su núcleo, reserva memoria para su uso interno y por ultimo se dispone a cargar un sistema de archivos en el cual pueda trabajar (nuestro Root File System).
- Aqui aparece otro protocolo de transferencia de datos llamado NFS (Network file system), ya a nivel usuario (root), el cual permitira al Kernel transferir hacia la maquina cliente y posteriormente, cargar en memoria todo un sistema raiz de archivos.
- Una vez transferido el sistema de archivos completo, Init se ejecuta y podemos tener la pantalla de login correspondiente.

(Inspirado en el articulo de Ken Yap – linuxfucocus.com)

ANEXO 2 ¿Que es DHCP?

DHCP: Dynamic Host Configuration Protocol (Protocolo de configuracion dinamica de host).

Es compatible hacia atras con BOOTP.
La RFC 2131 (antiguo RFC 1531), es la especificacion oficial.
Utiliza el puerto 67 de UDP

El servidor de DHCP es quien provee a las terminales de un IP y un nombre de archivo para descargar por medio de TFTP.

Opciones
Se puede configurar DHCP para pasarle parametros a una terminal, como el rootpath para montar el sistema de archivos, el IO de una placa de RED ISA, la ruta del archivo de menu para GRUB, etc.
Cada opcion DHCP tiene un nombre, un codigo y una estructura. El nombre es usado para referirse a la opcion. El codigo es un numero usado por el servidor y cliente DHCP para referirse a una opcion. La estructura describe el contenido de una opcion.
Para definir una nueva opcion, se necesita un nombre que no se use en otra opcion.
Los codigos pueden ser un numero entre 128 y 254. La estructu ra puede ser boolean, integer, ip-address, text, data string, encapsulation, arrays, records.

Ejemplo:

```
option   option-128 code 128 = string;
option   option-129 code 129 = text;
option   nbgrub-menu code 150 = text;
option   pxegrub-menu code 151 = text;
```

Ejemplo sencillo del archivo dhcpd.conf :

```
server-identifier       server;
default-lease-time      21600;
max-lease-time          21600;
use-host-decl-names on;
option subnet-mask               255.255.255.0;
option broadcast-address         192.168.1.255;
subnet 192.168.1.0 netmask 255.255.255.0 {
      range dynamic-bootp 192.168.1.110 192.168.1.118;
}

host ws01 {
            filename            "/lts/vmlinuz-2.4.21-ltsp-1";
            option root-path    "192.168.1.254:/opt/ltsp/i386";
            hardware ethernet   00:E0:7D:D6:EA:39;
            fixed-address       ws01;
}
```

Extraido del articulo “THIN CLIENT” de Joaquin Salvarredy <joaquinsalvarredy@yahoo.com.ar> 20 de mayo de 2004 .

ANEXO 3 ¿Que es TFTP?

TFTP: Trivial File Transfer Protocol.
Utiliza el puerto 69 de UDP.
Esta pensado para ser usado por sistemas diskless (normalmente workstations o X terminals). A diferencia de FTP, TFTP fue diseñado para usar UDP para hacerlo mas simple y mas pequeño.
La RFC 1350 es la especificacion oficial de la version 2 de TFTP.

Ejemplo :

```
joaquin@nika:~$ tftp localhost
tftp> get example.tftp
Received 14 bytes in 0.0 seconds
tftp> quit
```

Seguridad
Notar que los paquetes se enviaron sin pedir nombre de usuario o password. Esto es una caracteristica de TFTP. Recordar que TFTP fue diseñado para usarse durante el proceso de booteo de terminales con lo cual es imposible proveer un usuario y una contraseña.
Esta caracterrıstica de TFTP fue usada por muchos cracker para obtener copias de los archivos de password y luego entrar con cuentas guess. Para prevenir este tipo de acceso, la mayoria de los servidores TFTP proveen una opcion para bajar solo los archivos de un directorio especifico, por ejemplo `/tftpboot` en los sistemas Unix. Este directorio contiene solo los archivos de arranque requeridos por el sistema diskless.

Configurando TFTP
La configuracion del servidor TFTP es bastante sencilla, por lo general se agrega al metaservicio inetd o Xinetd
INETD
Hay que agregar en el archivo /etc/inetd.conf, la siguiente linea :

```
tftp dgram udp wait root /usr/sbin/tcpd /usr/sbin/in.tftpd -s /tftpboot
```

En el directorio `/tftpboot` estaran las imagenes de los kernels para descargar.
El directorio debera tener los permisos `chmod 555` pues algunos servidores no los enviaran si no se puede leer el directorio.
XINETD
En el caso de usar Xinetd se debera agregar en el directorio `/etc/xinetd.d` un archivo llamado `tftp` con el siguiente contenido:

```
 #  description: The tftp server serves files using the trivial
 #  file transfer protocol. The tftp protocol is often used to
 #  boot diskless workstations, download configuration files to
 #  network-aware printers, and to start the installation
 #  process
 #  for some operating systems.
service tftp
{
          disable       = no
          socket_type   = dgram
          protocol      = udp
          wait          = yes
          user          = root
          server        = /usr/sbin/in.tftpd
          server_args   = -s /tftpboot
}
```

Extraido del articulo "THIN CLIENT" de Joaquin Salvarredy <joaquinsalvarredy@yahoo.com.ar> 20 de mayo de 2004 .

ANEXO 3b TFTP, otra aproximacion...

TFTP (Trivial File Transfer Protocol) es un protocolo de transferencia de archivos, como lo es ftp, pero mucho más simple, a fin de que quepa en las EPROMs. TFTP puede ser usado de dos maneras:
tftp simple:

> El cliente puede acceder a la totalidad de su sistema de archivos. Es simple pero puede generar un gran problema de seguridad (cualquiera puede obtener su archivo de claves vía tftp).

tftp seguro (-s):

> El servidor `tftp` utiliza un llamada de sistema `chroot.2` para cambiar a su propio directorio raíz. Cualquier cosa que esté fuera del nuevo directorio raíz es completamente inaccesible, ya que el directorio que sufre el chroot se transforma en el nuevo directorio raíz; el campo `hd` del archivo `bootptab` debe reflejar esta nueva situación.
>
> Por ejemplo: cuando se utiliza `tftp` simple, el campo `hd` contiene el *path* o camino completo del directorio de arranque: `/export/root/machine1`. Cuando se utiliza `tftp` seguro con `/export` como directorio raíz, entonces `/export` pasa a ser `/` y el campo `hd` debe ser `/root/machine1`.

Asimismo, cada implementación Unix contiene el servidor `tftp` y, probablemente, no sea necesario que Usted instale su propio servidor.

ANEXO 4 ¿Que es NFS?

El Network File System (NFS) es un protocolo de transferencia de archivos que permite montar una particion remota en una maquina Linux como si fuera un recurso local, de esta forma tenemos acceso rapido y fiable por sobre todas las cosas, ya que la comunicacion es síncrona, o sea que no se libera el recurso hasta no haber terminado de escribir en el.
NFS se divide en dos partes, el Servidor y el Cliente.
Actualmente NFS evoluciono en 3 versiones: NFSv2 se encuentra en los Kernels de la serie 1.2.* hasta la serie 2.2.*, NFSv3 en los Kernels 2.4.* en adelante y NFSv4 en la serie 2.6.*.
La version 3 es la mas difundida y es la que pretendo tratar en este anexo. Una de las ventajas de esta version es que permite la transferencia de archivos mayores a 2GB.
Configurando el Servidor
La configuracion se divide en dos partes:

- Dar de alta y modificar los archivos de configuracion.
- Reiniciar el servicio NFS

Hay 3 archivos de texto principales: /etc/exports, /etc/hosts.allow y /etc/hosts.deny.
En realidad con solo setear /etc/exports es suficiente para que el servicio de NFS funcione correctamente.

/etc/exports
Una entrada tipica en el archivo seria:

```
directorio maquina1(option11,option12)
directorio maquina2(option21,option22)
```

Donde:
`directorio:`
Es el directorio que queremos compartir. Todos los subdirectorios dentro de este directorio, tambien por defecto son compartidos.

`maquina1 y maquina2:`
Son las maquinas clientes las cuales van a tener acceso al recurso compartido. Pueden ser declaradas por su DNS name (maquina1 por ejemplo) o su direccion IP (192.168.0.101 por ejemplo).

`optionxx`
Las opciones listadas para las maquinas que podran acceder son las siguientes:

- ro = el directorio a compartir es de solo lectura, esta es la configuracion por defecto.
- rw = el directorio a compartir se podra modificar
- no_root_squash = por defecto, cualquier peticion de archivos hecha por root del cliente es convertida en una peticion de usuario tipo root del servidor, esto implica obviamente un riesgo de seguridad.
- no_subtree_check = Si una parte del volumen es exportada, se activa esta rutina para verificar que pertenca al volumen principal.
- sync = por defecto, el manejo de archivos es en forma asincrona, esto causaria corrupcion de datos si el servidor se reiniciara, sync, evita esto, permitiendo a NFS terminar de manejar los archivos.

Supongamos que tenemos dos maquinas clientes con direcciones IP 192.168.0.1 y 192.168.0.2 y queremos compartir archivos binarios y directorios /home con estas maquinas. La configuracion

de /etc/exports seria:

```
/usr/local     192.168.0.1(ro) 192.168.0.2(ro)
/home          192.168.0.1(rw) 192.168.0.2(rw)
```

Aqui podemos observar que /usr/local tendra permisos de solo lectura para 192.168.0.1 y para 192.168.0.2 y el directorio /home tendra permisos de lecto-escritura para ambas maquinas.
Tambien podemos declarar una red completa, si tenemos muchas maquinas clientes que deberan acceder a estos recursos. Se declara el rango de red seguido de la mascara correspondiente:

```
/usr/local     192.168.0.0/255.255.255.0(ro)
/home          192.168.0.0/255.255.255.0(rw)
```

Tambien podemos usar comodines, si por ejemplo tenemos varios tipos de redes: `*.foo.com` o `192.168.`

La segunda seccion solo merece comentar que para iniciar este servicio en las distros tipo RH, solo debemos escribir lo siguiente:

```
# service nfsd {start|stop|restart|status}
```

Usando alguna de las opciones aqui descriptas: start, stop, restart, o status

NFS otra aproximacion

Network File System es la manera nativa de compartir archivos en los sistemas like-unix. Fue desarrollado para permitir a una maquina montar particiones de otra maquina remota y trabajarlo como si fuera una particion local.
Pero NFS no es un protocolo demasiado eficiente y es muy lento para conexiones mediante modem. Esta diseñado para redes locales, siendo muy flexible.
Ofrece muchas posibilidades tanto a usuarios como a administradores.
Lo que comunmente se llama NFS esta formado por 4 protocolos distintos. Cada uno depende de las Remote Procedure Calls (RPC) y de portmap (tambien llamado rpc.portmap). Un portmapper convierte numeros de programa RPC en numeros de puerto. Cuando un servidor RPC se inicia, dice a portmap que puerto usara y el numero de programa RPC manejado. Cuando un cliente quiere enviar una peticion RPC a un numero de programa dado, primero contacta con el servidor portmap para tomar el numero de puerto dando acceso al programa deseado. Despuus, dirige los paquetes RPC al puerto correspondiente. Los 4 servicios que permiten funcionar a NFS son:

Protocolo	Descripcion	Demonio
nfs	Este protocolo es el basico y permite crear, buscar, leer o escribir ficheros. Este protocolo tambien maneja autentificacion y estadisticas de ficheros.	nfsd
mountd	Este se encarga de montar sistemas exportados para acceder a ellos con nfs. El servidor recibe peticiones como mount y umount debiendo mantener informacion sobre los sistemas de ficheros exportados.	mountd
Nsm (Network Status Monitor)	Se usa para monitorizar los nodos de la red y asi conocer el estado de una maquina (cliente o servidor). Informa, por ejemplo, de un rearranque.	statd
Nlm (Network Lock Manager)	Para impedir modificaciones de los datos por varios clientes al mismo tiempo, este protocolo maneja un sistema de bloqueo. Asi con la ayuda del protocolo Nsm es posible conocer cuando se esta reiniciando un cliente. Nsm libera todos los bloqueos del cliente antes de devolverlos.	lockd

Extraido del articulo "THIN CLIENT" de Joaquin Salvarredy <joaquinsalvarredy@yahoo.com.ar> 20 de mayo de 2004 .

ANEXO 5 ¿Que es SVGAlib?

SVGAlib es una biblioteca gráfica de bajo nivel para Linux. Esta refuerza al lenguaje C, el cual no suministra soporte para gráficos.
C solo ofrece texto, ya que las funciones graficas son dependientes del sistema operativo, o sea que las funciones graficas de Windows no sirven en absoluto en Linux, deben ser re-escritas.
La solucion para el uso de graficos en C en linux es SVGAlib, un conjunto de librerias nativas de linux, para la manipulacion de graficos.

SVGAlib Vs. Xwindow
El sistema Xwindow es un servidor, el cual debe ser previamente iniciado para poder utilizar cualquier aplicacion que requiera X, ademas Xwindow consume una importante cantidad de recursos de memoria y espacio, lo que lo hace poco viable para este tipo de implementaciones.

SVGAlib, no requiere ningun tipo de inicializacion ni ocupa recursos, ya que es una libreria escrita en C que solamente es llamada cuando una rutina o programa necesite usar graficos.

(Inspirado en el articulo de Jay Link – linuxfocus.com)

ANEXO 6 ¿Que es un disco RAM o RAMDISK?

Un disco RAM es una parte de la memoria fisica de la computadora que puede utilizarse como si fuese una particion de disco.
Basicamente consiste en reservar cierta cantidad de memoria y accederla como si fuese una unidad de disco.
Se utiliza por ejemplo para cargar todo un Sistema Operativo, tal cual es la filosofia de los clientes finos, o cuando los archivos a los cuales se va a acceder se utilizaran de forma intensiva, mejorando la performance de la lectura y escritura, ya que la memoria RAM es mucho mas rapida que un disco duro.
(Inspirado en el articulo de Mark Nielsen – linuxfocus.com)

ANEXO 7 Breve guia de referencia a vi

MOVIMIENTOS DEL CURSOR

```
    ^
    k                   Indicación: La tecla h está a la izquierda y mueve a la izquierda.
< h     l >             La tecla l está a la derecha y mueve a la derecha.
    j                   La tecla j parece una flecha que apunta hacia abajo.
    v                   La tecla k mueve el cursor hacia arriba.

[Inicio]        =       El cursor se mueve al principio de la linea.
[Fin]           =       El cursor se mueve al final de la linea.
[RePag]         =       Mueve el texto una pagina hacia arriba
[AvPag]         =       Mueve el texto una pagina hacia abajo.
```

Vi posee dos modos: Comandos e Insertar

Modo 'Comandos':
Presiona la tecla (esc) para cerciorarte que estas en el modo 'Comandos', si no estas seguro si estas o no en este modo, presiona repetidamente la tecla (esc) hasta que escuches un pitido por el altavoz de tu CPU.
En modo Comandos, usamos los ':' precedidos de una o mas letras.

Algunos comandos faciles de recordar:

```
(:q)                    =               Salir de Vi
(:w)                    =               guardar documento
(:w nombre_archivo)     =               guardar documento con nombre 'nombre_archivo'
(:wq)                   =               guardar y salir de Vi
(:q!)                   =               Forzar la salida de Vi
(Ctrl+g)                =               Muestra el estado del documento al pie de la pagina.
(/ cadena_de_caracters + n o + N ) =    Busca una cadena de caracteres en el documento, luego con 'n'
repite automaticamente la busqueda hacia abajo o con 'N' hacia arriba.
(:help)                 =               ayuda en linea.
```

Modo 'Insertar':
Este es el modo de edicion propiamente dicho, una vez ingresado a este modo podemos modificar el contenido del texto en si. Presiona la tecla (esc) para cerciorarte que estas en el modo 'Insertar', si no estas seguro si estas o no en este modo, presiona repetidamente la tecla (esc) hasta que escuches un pitido por el altavoz de tu CPU.

En el modo 'Insertar' usamos letras sin ':', he aquí algunos comandos faciles de recordar:

(a) = Entramos al documento, listos para modificarlo, fijate que en el pie de pagina aparece la palabra 'INSERTAR'.
(i) = Entramos al documento, listos para modificarlo, fijate que en el pie de pagina aparece la palabra 'INSERTAR'.
(x) = Borra un carácter.
(dw) = Borra una palabra entera
(d$) = Borra una linea entera
(dd) = Borra una linea entera
([nn]dd) = Borra [nn] lineas enteras hacia abajo. Ejemplo: 2dd = borra dos lineas completas.
(u) = Deshace los cambios
(p) = pegar los datos almacenados en el buffer por el uso de (x), (dw), (d$) o (dd).
(r) = reemplaza un carácter. Una vez pulsado 'r' sobre el carácter a reemplazar, tipear el nuevo.

ANEXO 8 Virtual Network Computing o VNC ¿que es?

A principios de 1999 AT&T compró el Olivetti Research Laboratory y prosiguió con el desarollo de VNC. Como explican en el site de AT&T, el apellido VNC viene del desarollo de "clientes finos", es decir computadoras de redes (NC) funcionando según la tecnología ATM (Asynchronous Transfer Mode) y "puesto que el cliente VNC (visor) es una versión software de éste NC por ATM, permite la creación o la supresión de estaciones de trabajo al vuelo, hemos llamado a éste sistema Virtual Network Computing (Red virtual de Computadoras)".
Para usar VNC, necesitan una conexión TCP/IP, un servidor VNC (vncserver) y un cliente VNC (vncviewer) para conectar con la máquina en la cual el servidor VNC está activo. El servidor VNC basado en X funciona según la arquitectura cliente-servidor del mismo X. De hecho obtienen dos servidores en uno: un servidor X y un servidor VNC. Eso permite conservar el mismo número de visualización para el servidor X y el servidor VNC. Para conectar a un servidor VNC, tienen que dar el apellido de la máquina y el número de visualización. El comando parece a "vncviewer nombredemáquina:2". Es decir, si el servidor X tiene el número 2, el servidor VNC también tiene el número 2. Desde el cliente, el viewer, pueden conectar al servidor usando éste número. Si un usuario trabaja en la visualización número 1 y se conectan en el número 2, no se dará cuenta que trabajan en su máquina. VNC es capaz de encontrar el primer número de visualización disponible e informar, pero nada impide usar a otro número. Si vncserver propone el número 2, pueden conectar en la visualización 3,4... si quieren éste número al arrancar vncviewer.

VNC existe para numerosos SO diferentes, a veces únicamente como cliente. Es decir, sólo el viewer está disponible. Es el caso de BeOS. Muchos Unices, MacOS, AmigaOS... pueden usar VNC.

Lado servidor
La versión actual de VNC es 3.3.3 con números de revisión diferentes según el sistema. En las máquinas Unix (o usando X) obtienen un programa llamado vncserver y otro llamado Xvnc. Vncserver es un script Perl y pueden modificarlo según sus necesidades. Sirve para arrancar Xvnc, que es mejor evitar arrancar directamente. Estos programas pueden ser instalados en donde quieren, siempre que el directorio se encuentre en el camino de acceso. Cualquier gestor de ventanas (window manager) puede ser utilizado si está definido por defecto en el script xstartup (bajo el directorio `.vnc`). Arrancar vncserver por primera vez pide por un password, que necesitarán para conectar a éste servidor. Como cada servidor X, vncserver posee muchas opciones. Si teclean Xvnc --help tendrán una lista completa de las mismas. Si usan estos argumentos, vncserver los pasará a Xvnc, que los usa de manera preferente.

¡Es todo lo que necesitan para utilizar el servidor VNC!

Lado cliente
El cliente se compone de un ejecutable llamado vncviewer. Para conectar a un servidor VNC, basta con arrancar vncviewer indicando el número de visualización. Por ejemplo si quieren conectar a un servidor llamado linux en la visualización número 2, sólo tienen quenúmero 2, sólo tienen que escribir "vncviewer linux:2". El password del servidor les será pedido y después se encontrarán en el desktop de la máquina linux como si trabajaban en él. Si están como root, pueden administrar perfectamente la máquina. De modo que cuidado ¡Mejor saber lo que hacer! Todo eso es cierto para cada cliente en cada plataforma.

¡Sólo una palabra: fantástico!

(Extracto del articulo de George Tarbouriech – linuxfocus.com)

ANEXO 8b VNC, otra aproximacion...

Extraido del Articulo "Virtual Network Computing" de la revista Internet Computing – año 1998
Autores: Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood y Andy Hopper.

Debajo de VNC encontramos un protocolo remoto de visualizacion muy simple, es esta simplicidad lo que lo hace tan potente.
A diferencia de otros protocolos remotos de visualizacion como Xwindow o Citrix, VNC es totalmente independiente del sistema operativo.

Los origenes de VNC
Xwindow permite mostrar una interface grafica a una maquina remota. Olivetti & Oracle Research Laboratory extendio esta funcionalidad a un sistema que llamaron "Teleporting System" permitiendo a una interface de usuario que esta corriendo una aplicacion X remota, dinamicamente redireccionarse a diferentes displays.
Las ventajas de este nuevo sistema "Teleporting" eran muchas con respecto a Xserver:

- ➔ X requiere que la maquina cliente corra un servidor X. Esto requiere sustancialmente muchos recursos del cliente, el cual muchas veces por su arquitectura, le es imposible entregar.
- ➔ El modelo de seguridad de X hace inherentemente peligroso el permitir a una maquina remota usar tu pantalla. De forma generalizada, los administradores de grandes redes paran el trafico de X entrante o saliente a sus sites.
- ➔ El arranque de la aplicacion X es extremadamente lento y con mucha latencia debido a las llamadas de sistema de una aplicacion tipica.
- ➔ Encima el nombre "Xwindow" se parece mucho a un producto propietario de la empresa Microsoft.

1994 - VideoTile: el cliente ultra fino de Olivetti & Oracle Research Laboratory.
En 1994 Olivetti & Oracle Research Laboratory construyo "Videotile" como experimento en tecnologias de clientes finos. El aparato era en si una pantalla LCD con un lapiz optico y una conexion de red ATM.
Al principio habia sido diseñada para poder mostrar video con buena calidad, pero poco tiempo despues el perfil cambio para poder interactuar con otras aplicaciones.
El primer experimento fue enviar un video de una interface grafica, esto funcionaba muy bien pero consumia mucho ancho de banda.
Aplicando un poco mas de inteligencia a la aplicacion, la rediseñaron y ahora solo enviaba la informacion que se modificaba en pantalla, o sea solo las partes que cambiaban, esta fue la idea principal del protocolo VNC.

1995 – Accediendo a traves de un explorador con JAVA
Cuando SUN realizo la version ALFA de Java alla por 1995 y su explorador HotJava en el 95, Olivetti & Oracle Research Laboratory hizo la implementacion de Videotile para acceder a aplicaciones mediante la Web. ¡Se escribio todo el codigo del nuevo cliente en un dia y el .class resultante era de 6Kb nada mas!
Ahora, cualquier browser con capacidades java podia acceder a un escritorio de usuario y sus aplicaciones de forma remota a gran escala.

El protocolo VNC
Debajo del sistema VNC lo que encontramos es un protocolo simple de acceso remoto a interfaces graficas. Trabaja en el nivel de FrameBuffer, el cual es independiente de cualquier visualizador

grafico, cualquier aplicacion y cualquier sistema operativo, acoplandose a cualquier dispositivo que tenga capacidad de comunicacion del tipo TCP/IP.
La punta de usuario es el “cliente o visor VNC”, la punta donde se origina la informacion compatible con el FrameBuffer, la cual transporta el grafico, es el “Servidor VNC”.
El principio basico de visualizacion de este protocolo se parece a esto:
“poner un rectangulo en la posicion X,Y”.
Esto a primera vista pareceria bastante ineficiente el hecho de tener que redibujar constantemente, pero en la practica no es asi, ya que depende de la velocidad de las comunicaciones, el ancho de banda, el esquema de codificacion y la capacidad de proceso de ambas maquinas.
El mas bajo comun denominador es la llamada “codificacion RAW”, donde los datos de los pixeles de un rectangulo son simplemente enviados de izquierda a derecha en forma de lineas.
Todos los clientes y servidores VNC deben soportar esta codificacion, pero de todas formas, las capacidades de codificacion actualmente pueden ser negociadas entre el cliente y el servidor segun sus capacidades y su conexion.
Por ejemplo, la codificacion de la copia de un rectangulo es muy eficiente, y puede ser usada cuando el cliente tiene los mismos datos de pixel dando vueltas por todo el FramaBuffer, este tipo de codificacion se utiliza cuando por ejemplo el usuario mueve una ventana a lo largo y ancho de la pantalla, solo refrescando ciertas areas, los bordes por ejemplo.
Una forma curiosa de agilizar el proceso de codificado de imagenes por VNC en los clientes, es simplemente, intentando tener un solo color uniforme como fondo de pantalla y evitar dibujos.

En resumen:
El protocolo se divide en:

Sevidor VNC: en el servidor linux/unix encontramos a Xvnc, el cual es un servidor X11 con un sistema de vision remota incorporado.

Cliente VNC: el cliente debe estar soportado por la arquitectura de la terminal.
La aplicacion (firefox, openoffice) corre localmente en el servidor, el cliente VNC hace solicitudes de refresco de pantalla regularmente. Esta operacion puede resultar en redibujar la pantalla completa.

ANEXO 9 Etherboot en disco duro

Generalmente, las rutinas Etherboot se utilizan en chips ROM que se insertan en las placas de red o en diskettes para simular el acceso a ROM, pero en estos años recibi muchas preguntas sobre como aprovechar discos viejos para tal fin, ya que los chips pueden llegar a costar fortunas (hasta U$s30) y el verdadero 'talón de Aquiles' de los sistemas thinclient son los diskettes que actualmente tienen una calidad pesima.
Dado este panorama, la mejor forma es adquirir un disco duro viejo de 50MB o 100MB (o lo que encuentres... podriamos hablar hasta de 1MB!) e instalarle una rutina Etherboot. De esta forma evitariamos para siempre el uso de la diskettera y el consabido dolor de cabeza que traen actualmente los diskettes, debido a su malisima calidad. Ademas, convengamos que cuando utilizamos una maquina vieja para convertirla en cliente fino, la mayoria de las veces nos encontramos tambien con un disco duro dentro, ese vamos a aprovechar!
Items necesarios:

- Un diskette formateado y que por lo menos tengo 64KB de espacio disponible.
- Un disco de arranque 'Tom's Root Boot Disk' (en google lo encontramos)
- Una conexion a Internet
- Conocimientos de como usar la plicacion 'fdisk' de Linux.

Paso 1: Bajar la imagen correcta perteneciente a tu placa de red.

- Para esto debes fijarte en el chipset de la placa, generalmente encontramos algo como 'RTL8139', que es uno de los chipsets mas usados en el mercado.
- Deberias entrar a www.rom-o-matic.com, te recomiendo la relese 5.2.2, ya que es con la cual yo trabajo hace años sin ningun tipo de percances.
- Elije el chipset y obtiene una imagen del tipo 'LILO Bootable ROM (.zlilo Image) en el menu despleglable, luego hace clic en el boton 'Get ROM!'
- Ahora copia normalmente en el diskette esta imagen .zlilo, primero montando el diskette y luego usando el comando regular 'cp'. Desmonta el diskette, no te olvides!

Paso 2: Accion de 'fdisk' en el disco duro de la maquina cliente.

- Bootea la estacion con el diskette 'Tom's Root Boot Disk', cuando te pida que saques el diskette no lo hagas, lo vas a necesitar, ahora ingresa como 'root' siguiendo las instrucciones en pantalla.
- Ejecuta 'fdisk' > # fdisk /dev/hda
- Usa 'fdisk' para limpiar TODAS las particiones del disco y crear una unica particion nativa Linux. No importa el tamaño de la particion, ya que vas a usar muy poco de ella.

Uso de Fdisk:

- Tipea 'm' para ver el menú
- 'p' te mostrara todas las particiones que tiene el disco
- cuando uses 'd' para borrar las particiones, borralas de la ultima hasta llegar a la primera.
- Usa 'p' de nuevo para verificar que no han quedado particiones activas
- ahora usa 'n' para crear una nueva particion, haz que la particion sea unica, primaria y nativa de linux (etx2)

- Guarda los cambios y sal de 'fdisk'

Paso 3: Crear un sistema de archivos en el disco duro y montarlo

- Crea un sistema de archivos en la particion que definiste > # mke2fs /dev/hda1
- crea un directorio /boot en tu RAMDISK (en memoria) > mkdir /boot
- Monta el disco duro dentro de /boot > mount /dev/had1 /boot
- monta ahora el diskette 'Tom's Root Boot Disk' dentro de /mnt > mount /dev/fd0 /mnt

Paso 4 Copiar los archivos de LILO al disco duro

- copia el archivo 'boot.b' de LILO al disco duro > cp /mnt/boot.b /boot
- copia el directorio 'map' de LILO al disco duro cp /mnt/map /boot
- desmonta el diskette 'Tom's Root Boot Disk' > umount /dev/fd0
- saca el diskette 'Tom's Root Boot Disk' y coloca el diskette con la imagen de tu placa de red
- monta el diskette > mount /dev/fd0 /mnt
- lista el diskette > ls /mnt
- copia la imagen de tu placa de red al disco duro > cp /mnt/nombre_de_la_imagen /boot

Paso 5: Crear el sector de arranque (Boot Sector) del disco duro

*NOTA: ***¡Antes que nada, el comando a ejecutar a continuacion es bastante largo, asi que por favor, revisa bien la sintaxis y no te comas ni espacios ni barras! (a continuacion de '\' presiona 'Enter' asi continuas en la linea de abajo).****

```
echo image=/boot/nombre_de_la_imagen label=UTC | /usr/bin/lilo \
-C - -b /dev/hda -i /boot/boot.b -c -m /boot/map
```

Vas a obtener un breve mensaje de LILO, en este momento, podes reiniciar la estacion de trabajo y simplemente... ¡funciona!

Si no funciona, deberas llenar de ceros el primer megabyte del disco duro, para eso, bootea con el 'Tom's Root Boot Disk' y a continuacion tipea el siguiente comando:

```
# dd if=/dev/zero of=/dev/hda count=1000000
```

Ahora vuelve a comenzar desde el paso 2... ¡Suerte!

ANEXO 9b EtherBoot en Disco duro serializado

En el anexo anterior explique paso a paso como crear la marca de arranque de la placa de red correspondiente a cada cliente en un disco duro obsoleto, pero, ¿que pasaria si tuviesemos 20 terminales y consiguieramos 20 discos viejos?. ¿Cuanto tiempo tardariamos en implementar la solucion a cada disco?.
Mucho tiempo ¿verdad?.
Por suerte hay una microdistribucion basada en la distribucion “Tom Root Boot Disk” con soporte fisico en diskette que nos puede facilitar enormemente esta tarea.
La gente de Wizzy Digital Courier ha desarrollado esta herramienta a la que llamaron “wizzy-tomsrtrb”.
Basicamente hace todo el trabajo por nosotros con la sola ejecucion de un script, pero tiene un extra que lo hace mas valioso: trae incorporada la base de todas placas placas de red soportadas por etherboot, de esta manera, una vez ejecutado el script, ¡tendremos una marca de arranque en disco UNIVERSAL para cualquier tipo de placa existente!.
Pasos a seguir:

- ➔ Bajamos el tarball de ftp://ftp.wizzy.com/pub/wizzy/wizzy-tomsrtbt-2.0.105.tar.gz
- ➔ Descomprimimos el archivo y ejecutamos en el directorio contenedor: `$ ./install`, si no nos da permiso de ejecucion, ingresemos como superusuario y repitamos la operacion.
- ➔ Seguir los pasos de creacion del diskette serializador.
- ➔ Encendemos la maquina cliente.
- ➔ Insertamos el diskette recien creado y una vez que nos de el control (leer en pantalla), tipeamos: `# ./ltsp.sh /dev/hda1`
- ➔ El script crea una particion tipo Linux e instala LiLo con la marca de arranque EtherBoot.

¡Buena suerte!

ANEXO 10 Kernel cmdline options (comandos al vuelo para el Kernel, como funcionan)

Solo con Etherboot

Para pasarle al Kernel lineas de comando al vuelo, necesitamos especificar un par de opciones antes en el archivo dhcpd.conf
La forma típica es como sigue:

```
option   option-128   e4:45:74:68:00:00;   (*)
option   option-129   "KEYWORD=VALUE";
```

Donde KEYWORD es la opcion ue trataremos de enviarle al Kernel y VALUE es obviamente el valor que le daremos a KEYWORD.
Por ejemplo, para decirle al Kernel que use una placa de red especifica, podriamos hacer lo siguiente:

```
option   option-128   e4:45:74:68:00:00;
option   option-129   "NIC=3c509";
```

Podemos especificar multiples parametros al Kernel solamente separandolos con un espcacio.
Por ejemplo, si queremos decirle al Kernel que use cierta placa de red y ademas que configure el Framebuffer de video en modo consola, poriamos hacer lo siguiente:

```
option   option-128   e4:45:74:68:00:00;
option   option-129   "NIC=3c509 vga=773";
```

Con los ISC DHCP v3.0 y posteriores, necesitamos los tipos de opciones, ya sean texto o cadenas.
Esto se especifica al principio del archivo de configuracion de dhcpd*:*

```
option option-128 code 128 = string;
option option-129 code 129 = text;
```

Podemos agregar al modulo NIC usando directamente la opcion 129 al vuelo.
NOTA: Cada driver tiene su propio set de parametros posibles para el Kernel y opciones, busca informacion sobre tu placa de red en Internet.

```
option   option-128   e4:45:74:68:00:00;
option   option-129   "NICOPT=options=5"; * Ajusta el Modulo Tulip Media-Type a 100baseTx-FD
```

(*) En los ejemplos de arriba constantemente vemos junto al la opcion 128 el valor "`e4:45:74:68:00:00`". ¡Este valor NO ES una direccion MAC!.
Cuando usamos Etherboot, si queremos enviar el contenido de la opcion 129 al Kernel, deberemos habilitar la opcion 128 con `e4:45:74:68:00:00`.
Esta es una caracteristica especial que posee Etherboot, el cual entiende que a partir de ese momento le debe enviar al vuelo parametros al Kernel.

ANEXO 11 Configurando XDMCP (X Desktop Manager Control Protocol)

Lo mas facil de la configuracion de estos Servidores, ya sean para UTC o LTSP:
Vamos a:

```
Aplicaciones >
          Sistema >
               Configuracion >
                    <Configuracion de pantalla de entrada>
                    (Login Screen Setup)
```

Luego pestaña “general” y habilitamos un tema para la entrada remota.
Vamos a “seguridad” y comprobamos que XDMCP este habilitado.
En la pestaña “XDMCP” le decimos cuantos clientes vamos a colgar a gdm en “Maximo de sesiones remotas”.

ANEXO 12 Cosas a tener en cuenta antes de instalar LTSP

Extraido de mi articulo publicado en www.ubuntu-es.org *el dia 09 de Mayo de 2007 "Cosas a tener en cuenta antes de instalar LTSP".*

Hace tiempo que recibo via mail o por messenger dudas y preguntas sobre como instalar e implementar LTSP desde muchos lugares de habla hispana.
La mayoria de las implementaciones son para establecimientos educativos de toda indole.
Pero lo que mas me llama la atención es que al final, las preguntas se terminan repitiendo, terminan siendo las mismas y lamentablemente son las mismas porque no se encara adecuadamente la labor de implementacion de un entorno LTSP.
Recibo preguntas de gente que dice saber lo minimo e indispensable de GNU/LINUX para poder encarar un proyecto como LTSP, pero al leer sus preguntas, me doy cuenta que no es asi.
No podremos encarar responsablemente un proyecto en LINUX si ni sabemos como crear un directorio por ejemplo...
Ya sea si instalamos LTSP mediante una distro que lo traiga pre-instalado o nos embarcamos en la maravillosa experiencia de implementarlo en un sistema LINUX desde cero, antes que nada la persona responsable debe conocer por lo menos en un estado basico la estructura de directorios en LINUX, crear directorios y archivos vacios, moverse entre ellos, copiar, borrar, renombrar archivos y directorios, interpretar algunos logs de sistema, instalar binarios y no-binarios en su sistema anfitrion, configurar una placa de red e Internet, manejar un editor de textos, AMIGARSE CON LA TERMINAL!!!...

Luego en segunda instancia, concientizarse que nuestra BIBLIA se encuentra en el SITIO OFICIAL DE LTSP, que si alguno no lo sabe es www.ltsp.org.
Alli encontramos la DOCUMENTACION OFICIAL del proyecto y ademas muchisima colaboracion de la comunidad LTSP.
Recomiendo comenzar buscando documentacion de la version 3.0 en Castellano y de ahi en mas leer el resto .
Sumado a nuestra BIBLIA, debemos buscar en internet mas informacion acerca de LTSP. Veremos que encontraremos sinnumero de articulos relacionados con este proyecto, tanto en Castellano como en Ingles.
Para poder llegar a EXITO, debemos entender de modo basico por lo menos acerca de TCP/IP, DHCP, TFTP, NFS, XDMCP, Inetd o Xinetd, cableado UTP.

Y aqui es donde esta el problema:
Las preguntas pidiendome ayuda siempre se parecen y siempre pregunto con que manual o tutorial se estan guiando, leo si no lo conzco al material y lamentablemente, la mayoria de las veces, la solución a los problemas se encuentra en el mismo manual o tutorial!.
POR FAVOR, hay que leer los manuales!
Hay que detenerse y volver a leer si no se entendio algo especifico!
Solo despues de eso; y hablo despues de buscar por toda internet!; nos empecemos a preocupar y a pedir ayuda en las listas de correo o en los canales dedicados a LTSP por IRC.

Preguntas tales como "¿como configuro dhcp?", "¿porque no veo los puntitos al arranque de la terminal?", "¿que significa `NO ROOT PATH`?", "¿por que me sale: `ERROR TFTPd NO SUCH FILE OR DIRECTORY`?" "¿por que no linkea la red?" NO DEBEN SER PREGUNTADAS JAMAS!
Y MENOS TODAVIA ESTE TIPO DE CONSULTAS: "No me anda LTSP, que hago?" o "A mi me da error y no me arranca el entorno grafico, donde puede estar el problema?"

Ya que toda esa informacion, INCLUIDA SU SOLUCION se encuentra en la documentacion oficial de LTSP y centenares de documentos independientes en la misma INTERNET.

Termino dandome cuenta que mucha gente que me pregunta y consulta, ni siquiera se toma el trabajo de hojear los manuales, trayendo aparejado que muchas veces este maravilloso proyecto quede descartado por "dificil" o porque "no anda" y ademas haciendome perder tiempo, explicandoles que lean los manuales.

La verdad es que no quiero ser duro con nadie pero soy muy sincero, creo que la forma de hacer crecer este proyecto en Latinoamerica es arreglando estos puntos flojos.

Dejo algunos LINKS realmente UTILES:

www.ltsp.org
http://www.ubuntu-es.org/index.php?q=node/20079
http://kde.mirrors.tds.net/pub/ltsp/docs/instructions-3.0-es.html
http://terminales.hispalinux.es/tiki-index.php
3244...
http://www.champinet.com/download/doc/index.html
http://www.k12ltsp.org/
http://www.gentoo.org/doc/es/ltsp.xml
http://comunidad.molinux.info/index.php/LTSP_en_la_escuela_con_Molinux

ANEXO 13 Implementacion de un entorno LTSP desde cero en Ubuntu

Extraido de mi articulo "Mini TUTO (alpha todavia) Edubuntu 6.06 + LTSP 4.2.1" publicado en www.ubuntu-es.org *el 30 de Junio de 2006 (+de 11.000 lecturas a fines de 2009).*
Aqui uso como sistema anfitrion a Edubuntu 6.06, pero es perfectamente aplicable a cualquier distribucion GNU/LINUX en la actualidad.

Instalacion & configuracion paso a paso de LTSP 4.2.1 en Edubuntu Desktop.
Version 0.01

ANTES DE HACER NADA: BACK-UP DE TODOS LOS TRABAJOS DEL SERVER, COPIAR `/etc/dhcp.conf` y `/opt/lts/i386/etc/lts.conf` a un diskette.

***** OBVIAMENTE ESTO SOLAMENTE SI TENEMOS YA CORRIENDO UN LTSP*****

*NOTA: ***Solo explico aqui como configurar un Server LTSP, como esta es una version pre-oficial de mi tutorial, prometo dar cuenta de instalacion de clientes en las proximas actualizaciones.*
*Por que no usar el LTSP que trae ubuntu? porque lei que la mitad de las funciones estan soportadas y porque el ultimo LTSP puede soportar clientes con un minimo de 7MB de RAM! (yo tengo un monton, todavia no he hecho funcionar eso!).****

Procedemos

1º instalo edubuntu 6.06 Desktop en la maquina que destinamos para servidor, claro esta.

2º configuro mi host (recordar que este sera el server), el nombre de mi host sera "server", mientras que el dominio "ltsp". Para configurar el entorno de red, vamos a Sistema > Administracion > Red. Generalmente, por standarizacion, LTSP por defecto, toma como IP del servidor: 192.168.0.254, podemos usar el que querramos.

3º Si tenemos un proxy, podemos tipear a mano: `http_proxy=http://[IP_Proxy]:[Puerto]`, o dejarlo permanente en `/etc/environment`, luego, update apt (apt-get update). Si apt te da problemas, ir a `/etc/apt/apt.conf` y donde dice proxy entre comillas, especificar la IP del mismo, siempre precedida por "http://".

NOTA: este punto lo inclui, porque dependo personalmente de un proxy, aca donde trabajo.

4º Una vez que el entorno esta listo y tenemos Internet para bajar algunos paquetes, vamos a www.ltsp.org y hacemos clic en el link "*Current Release - LTSP-4.2*".
Esto nos llevara al WIKI de ltsp, el cual nos explica (en ingles) las nuevas caracteristicas del paquete LTSP 4.2.1, nos advierte que antes que nada, hay que tener en cuenta en hacer una instalacion nueva en un directorio diferente, si es que teniamos instalada una version anterior de LTSP.
Como es eso?. Si antes teniamos instalado todo en `/opt/ltsp`, ahora debemos hacer algo como esto: `/opt/ltsp_4.2`, por ejemplo.
Debemos bajar ahora el paquete "`ltsp-utils 0.25`" en el formato .deb para UBUNTU.
Instalamos ltsp-utils 0.25

```
# dpkg -i 'paquete_a_instalar.deb'
```

5º run ltsp-utils, me fijo que servicios tengo instalados y que no, por defecto, solo xdmcp esta instalado.
Si no tenemos buena memoria, anotemos en que estado se encuentran los servicios necesarios.

6º synaptic o apt: instalar solo por comodidad, `ltsp-server-standalone`, esto instalara los siguientes paquetes, los cuales no vienen por defecto en Edubuntu Desk:

`binutils` (paquete de programas para compilar archivos objeto y manipular binarios)
`debconf-utils` (utilidades relacionadas con los paquetes .deb)
`debootstrap` (crea un sistema debian desde cero)
`dhcp3-server` (servidor DHCP en su version 3 - Dynamic Host Configuration Protocol)
`esound-clients` (cliente del demonio ESd)
`libevent1` (libreria para llamada de eventos)
`libnfsidmap1` (libreria necesaria para la ejecucion de NFS)
`ltsp-server` (ambiente basico de trabajo para un server LTSP, creado para ubuntu, depende de dhcp externo)
`netkit-inetd` (demonio manager de conexiones entrantes via tcp o udp)
`nfs-common` (utilidades para nfs, lo que es para RH "nfs-utils")
`nfs-kernel-server` (parche para el kernel)
`portmap` (necesario para funcionamiento de NFS)
`syslinux` (usado para bootear un sistema linux, aca se usa como booteador PXE)
`tftpd-hpa` (server Trivial File Transfer Protocol mejorado)

7º verificamos como estan los servicios ejecutando nuevamente `ltspadmin` (siempre como root)
Vemos que estan todos instalados, pero dhcpd no esta corriendo, generalmente porque no esta configurado, nfs y xdmcp tampoco...

8º nos disponemos a instalar LTSP 4.2.1, vamos a configurar el instalador y una vez configurado, instalamos los paquetes, todo esto dentro de ltspadmin, por supuesto.
El sitio donde se encuentra es: http://ltsp.mirrors.tds.net/pub/ltsp/ltsp-4.2/

Estos son los paquetes que LTSP instala por defecto:

```
ltsp-ltsptree-1.13-0-i386.tgz 0.03mb 100%
ltsp-libusb-1.0-0-i386.tgz 0.02mb 100%
ltsp-glibc-1.0-1-i386.tgz 8.15mb 100%
ltsp-x-core-1.6-0-i386.tgz 20.77mb 100%
ltsp-freetype-1.1-0-i386.tgz 0.56mb 100%
ltsp-x-fonts-100dpi-1.6-0-i386.tgz 12.08mb 100%
ltsp-xinetd-1.2-0-i386.tgz 0.08mb 100%
ltsp-perl-1.0-0-i386.tgz 6.76mb 100%
ltsp-rdesktop-1.6-0-i386.tgz 0.22mb 100%
ltsp-gdb-1.1-0-i386.tgz 1.54mb 100%
ltsp-kernel-2.6.16.1-ltsp-2-0-i386.tgz 5.22mb 100%
ltsp-modules-2.6.16.1-ltsp-2-0-i386.tgz 8.19mb 100%
ltsp-zlib-1.0-1-i386.tgz 0.02mb 100%
ltsp-snmpd-1.1-0-i386.tgz 3.82mb 100%
ltsp-ltspfsd-0.4-0-i386.tgz 0.01mb 100%
ltsp-sysvinit-1.1-0-i386.tgz 0.01mb 100%
ltsp-bash-1.1-0-i386.tgz 0.33mb 100%
ltsp-audiofile-1.1-0-i386.tgz 0.07mb 100%
```

```
ltsp-ypbind-1.1-0-i386.tgz 0.02mb 100%
ltsp_nasd-1.2-0-i386.tgz 0.61mb 100%
ltsp-module-init-tools-3.3-0-i386.tgz 0.04mb 100%
ltsp-libvncserver-1.1-0-i386.tgz 0.07mb 100%
ltsp-lbuscd-0.4-0-i386.tgz 0.01mb 100%
ltsp-libpng-1.1-0-i386.tgz 0.19mb 100%
ltsp-udev-1.0-0-i386.tgz 0.14mb 100%
ltsp-nbd-1.0-0-i386.tgz 0.03mb 100%
ltsp-ssh-1.1-0-i386.tgz 2.05mb 100%
ltsp-haltsys-1.1-0-i386.tgz 0.00mb 100%
ltsp-localdev-1.3-0-i386.tgz 0.00mb 100%
ltsp-popt-1.2-0-i386.tgz 0.10mb 100%
ltsp-tcp_wrappers-1.1-0-i386.tgz 0.03mb 100%
ltsp-open-1.2-0-i386.tgz 0.00mb 100%
ltsp-pam-1.1-0-i386.tgz 0.12mb 100%
ltsp-aumix-1.1-0-i386.tgz 0.01mb 100%
ltsp-util-linux-1.1-0-i386.tgz 0.01mb 100%
ltsp-udev-rules-1.1-0-i386.tgz 0.01mb 100%
ltsp-busybox-1.1-0-i386.tgz 0.08mb 100%
ltsp-ltspinfod-1.2-0-i386.tgz 0.01mb 100%
ltsp-net-tools-1.1-0-i386.tgz 0.02mb 100%
ltsp-vidlist-1.5-0-i386.tgz 0.01mb 100%
ltsp-esd-1.1-0-i386.tgz 0.08mb 100%
ltsp-sane-1.0-0-i386.tgz 2.72mb 100%
ltsp-openssl-1.1-0-i386.tgz 1.69mb 100%
ltsp-prep_swap-1.1-0-i386.tgz 0.00mb 100%
ltsp-lp_server-1.2-0-i386.tgz 0.01mb 100%
ltsp-pci_scan-1.1-0-i386.tgz 0.00mb 100%
ltsp-startsess-1.2-0-i386.tgz 0.00mb 100%
ltsp-ncurses-1.1-0-i386.tgz 0.77mb 100%
ltsp-e2fsprogs-1.2-0-i386.tgz 0.70mb 100%
ltsp-x-fonts-1.6-0-i386.tgz 19.49mb 100%
ltsp-nfs-utils-1.1-0-i386.tgz 0.13mb 100%
ltsp-portmap-1.1-0-i386.tgz 0.02mb 100%
ltsp-getltscfg-1.3-0-i386.tgz 0.01mb 100%
ltsp-libgcc_s-1.0-1-i386.tgz 1.51mb 100%
```

Un truco: la primera vez que se instala LTSP 4.2.1 en un sistema, guarda un directorio llamado pkg_cache, en `/tftpboot`, el cual es una copia casi exacta del mirror donde se encuentra todo, digo casi exacta, porque lo unico que hay que incluir es el archivo "packages.list" (que se encuentra en la raiz del mirror) a este directorio.
Una vez hecho esto, podemos quemarlo en un cd al directorio completo, e invocarlo desde ltspadmin de la siguiente forma "`file://ubicacion_nueva`".
NOTA: he probado con 3 barras como debe ser, o por lo menos como yo conocia que deberia funcionar (///), pero no me dio resultado, asi que con dos barras en "file:" me fuciono...

9º una vez instalados, vamos a configurar un par de cosas para dejar aparentemente listo LTSP:

a) Configuracion de dhcpd, ltspadmin lo configura de tal forma que arranque con el sistema y aparte crea un dhcpd.conf de ejemplo (opcion 3 del instalador ltspadmin).

b) Habilitar el demonio nfsd (opcion 6 del instalador ltspadmin), aca te detecta que el S.O. esta basado en Debian.

c) Habilitar XDMCP (opcion 7 del instalador ltspadmin) habilitar gdm y permitir login remoto grafico

d) Por ultimo, crear los archivos de configuracion: hosts hosts.allow, exports y lts.conf

Una vez que hicimos estos pequeños cambios, salimos de ltspadmin y volvemos a entrar, ya que de esta forma se actualizan los datos, nos tendria que mostrar algo parecido a esto:

```
ltspcfg v0.16 The Linux Terminal Server Project
(http://www.LTSP.org)
```

```
Interface IP Address Netmask Network Broadcast Used
eth0 192.168.10.223 255.255.255.0 192.168.10.0 192.168.10.255 to
return to the main menu...
```

Vemos que todavia hay un par de servicios que no estan corriendo debidamente.

Bien terminemos con dhcp:
Una cosa que es bien curiosa y que me trajo dolor de cabeza una tarde completa es que el archivo dhcpd.conf que se usa para habilitar el server dhcp en edubuntu, no se encuentra donde todos esperariamos que se encuentre, o sea en /etc/dhcp3/dhcpd.conf sino en /etc/ltsp/dhcpd.conf
Para saber si esta corriendo bien dhcp tipeamos:

```
# /etc/init.d/dhcp3-server restart
```

Si nos da `[fail]` en el arranque (en la parada no me preocupo, ya que si no estaba corriendo, por defecto se queja!), es basicamente porque esta apuntando a una red distinta a la nuestra, por defecto, apunta a 192.168.0.0, yo por ejemplo, mi server lo tengo apuntado asi: 192.168.10.223.
Para probar que sea este el problema, reemplazamos los nros. IP por los correctos.
En la maquina cliente podemos ver que ya funciona dhcp sin problemas, solo que no encuentra la imagen del kernel para ser descargada.
Ahora, el siguiente paso, por lo menos el que yo hago es reemplazar `/etc/ltsp/dhcpd.conf` por `/etc/dhcp3/dhcpd.conf.sample`, ya que trae un par de cositas utiles que nos van a hacer falta si tenemos maquinas muy viejas con placas de red ISA. (yo las tengo!)

El archivo original cuando se instala en UBUNTU es asi:

```
authoritative;

subnet 192.168.10.0 netmask 255.255.255.0 {
range 192.168.10.250 192.168.10.254;
option domain-name "example.com";
option domain-name-servers 192.168.0.1;
option broadcast-address 192.168.10.255;
option routers 192.168.10.1;
option subnet-mask 255.255.255.0;

filename "/ltsp/pxelinux.0";
option root-path "/opt/ltsp/i386";
}
```

el archivo .sample:

```
# dhcpd.conf

ddns-update-style ad-hoc;

option subnet-mask 255.255.255.0;
option broadcast-address 192.168.10.255;
option routers 192.168.10.223;
option domain-name-servers 192.168.10.223;
option domain-name "your_domain.org"; # You really should fix this
option option-128 code 128 = string;
```

```
option option-129 code 129 = text;

get-lease-hostnames true;

next-server 192.168.10.223;
option root-path "192.168.10.223:/opt/ltsp/i386";

subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.100 192.168.0.199;
if substring (option vendor-class-identifier, 0, 9) = "PXEClient"
{
filename "/lts/2.6.16.1-ltsp-1/pxelinux.0";
}
else{
filename "/lts/vmlinuz-2.6.16.1-ltsp-1";
}
}
#
# If you need to pass parameters on the kernel command line, you
can
# do it with option-129. In order for Etherboot to look at option-
129,
# you MUST have option-128 set to a specific value. The value is a
# special Etherboot signature of 'e4:45:74:68:00:00'.
#
# Add these two lines to the host entry that needs kernel
parameters
#
# option option-128 e4:45:74:68:00:00; # NOT a mac address
# option option-129 "NIC=ne IO=0x300";
#
```

Como se puede observar, ltspadmin, practicamente ya configuro por nosotros dhcpd.conf, asi que copiamos este dhcpd.conf.sample a `/etc/ltsp` reemplazandolo por el viejo dhcpd.conf, otra cosa que he visto que el archivo dhcpd.conf en /etc/dhcp3 esta intimamente ligado a /etc/ltsp/dhcpd.conf, o sea que deben estar los dos presentes (...)
No olvidemos de poner el nombre de dominio correcto! (recomiendo ltsp, ya que por defecto, LTSP 4.2.1, genera un archivo de texto muy grande con los nombres de los hosts, son 253!).
Lo unico que debemos arreglar, si es que no estamos en la regla por defecto de red privada 192.168.0.0 es justamente en "subnet w.x.y.0" y en "range w.x.y.z w.x.y.z+n".
Otra cosa, el kernel que se baja ltspadmin es el `vmlinuz-2.6.16.1-ltsp-2`, ojo con esto! ya que por defecto, /etc/ltsp/dhcpd.conf marca el `vmlinuz-2.6.16.1-ltsp-1`.

tftpd-hpa

El server ftp trivial tiene solo un par de cositas para ser modificadas, la primera y mas importante:

De por si, tftpd (hpa) trabaja en modo -s (seguro) y su directorio de booteo apunta exactamente a `/var/lib/tftpboot`, ahora bien, si queremos respetar el estandard de LTSP, este directorio en realidad se debe encontrar en "/" o sea que debera ser `/tftpboot`.
vamos a `/etc/default/tftpd-hpa` y reemplazamos el path por el correspondiente sugerido por LTSP, o de lo contrario, migramos todos los archivos que por defecto instalo ltspadmin al directorio por defecto de tftpd-hpa (`/var/lib/tftpboot`).

Luego le decimos que queremos que corra el demonio con "`yes`" en la sentencia '`RUN_DAEMON`'
Lo proximo es ir a `/etc/inetd.conf` y cambiar, donde esta el parametro de tftp, al final de la linea, el path correspondiente, en este caso "`/tftpboot`". Hay que recordar que en modo seguro, tftp, toma tal cual el parametro del path.

NFS

Vamos a `/etc/exports` y cambiamos la linea `/opt/ltsp` por `/opt/ltsp/i386`, cabe recordar que debe estar igual tambien en /etc/ltsp/dhcpd.conf, en la opcion root-path de lo contrario se detiene el sistema con un "`kernel panic`".
Luego, como root, creamos el path completo: `/var/opt/ltsp/swapfiles`, en Edubuntu solo llega a `/var/opt`.
Este directorio es unicamente usado si tenemos maquinas muy viejas, a las cuales les vamos a crear un sistema de cacheo o memoria virtual, tambien llamada memoria '`swap`' para que funcionen mas estables, particularmente lo uso con LTSP 4.0 porque tenemos maquinas muy viejas aqui.

Ahora reiniciamos NFS asi:

```
# /etc/init.d/nfs-kernel-server restart
```

Nos debe decir algo asi:

```
* Stopping rpc mountd...                                [ ok ]
* Stopping rpc nfsd...                                  [ ok ]
* Unexporting directories for NFS kernel daemon...      [ ok ]
* Exporting directories for NFS kernel daemon...        [ ok ]
* Starting rpc nfsd...                                  [ ok ]
* Starting rpc mountd...                                [ ok ]
```

Por ultimo, nos fijamos si esta funcionando con:

```
# showmount -e
```

Nos deberia dar algo como esto:

```
/opt/ltsp                     192.168.10.0/255.255.255.0
/var/opt/ltsp/swapfiles       192.168.10.0/255.255.255.0
```

Por ultimo, si la IP de nuestro server es diferente a la recomendada por defecto (192.168.0.254), debemos incluirla en `/etc/hosts` ya que si checamos con detenimiento, veremos que nuestra IP (en este caso la mia es 192.168.10.223, ya que el administrador del piso me dio esa) no esta en ningun lado, asi que al principio de /etc/hosts ponemos [IP=w.x.y.z] server.ltsp server

Reiniciamos el servicio inetd:

```
# /etc/init.d/inetd restart
```

volvemos a checar los servicios con ltspadmin, veremos algo parecido a esto:

```
ltspcfg v0.16 The Linux Terminal Server Project
(http://www.LTSP.org)
```

```
Interface IP Address Netmask Network Broadcast Used
eth0 192.168.10.223 255.255.255.0 192.168.10.0 192.168.10.255 to
return to the main menu...
```

Vemos que solamente falta XDMCP para que todo funcione correctamente, ya se puede probar el cliente, lo que veremos es una pantalla gris con una X en el medio, señal de que no nos funciona XDMCP!

XDMCP

Vamos a `Sistema > Administracion > Ventana de entrada`

Despues de pedirnos clave, aparecera una ventana, la cual permite configurar muchas cosas, entre ellas el acceso remoto grafico!

Hacemos clic en la pestaña "Remota" y seleccionamos "`Igual que la entrada local`", luego hacemos clic en el boton "`Configurar XDMCP`" y ajustamos cuantas sesiones queremos que acepte, cuantas al mismo tiempo, cuantas en cola de espera, por defecto, de por si los valores que trae son usables para una red pequeña de hasta 16 hosts.

EXITO! ACA TERMINA LA AVENTURA! EN ESTE MOMENTO ESTARIA FUNCIONANDO SIN MAS LTSP 4.2.1 CON EDUBUNTU!

*NOTA: ***Recordar que en este texto solo hablo del servidor, ya que se supone que teniamos funcionando LTSP en otros sistemas (RH, MANDRAKE, etc)****

ANEXO 14 Open SuSe 10 y LTSP desde cero:

Lun 14 May 2007

Banco de pruebas:
Hp Compaq d330 DT - 256MB - 40GB disco
Pentium 1 133 MMX - 16MB

Implementacion: LTSP 4.1.1

NOTA: UTILICE TAMBIEN EL MANUAL DE LTSP EN SUS VERSIONES 3 & 4

> Arrancamos con el CD 1 elegimos "Instalacion"

> El CD ejecuta sus aplicaciones de diagonostico y comienza la instalacion:

Idioma: elegimos : "ESPAÑOL". Clic en "SIGUIENTE".
Comprobacion de medios: comprueba el estado de los CDs es una herramienta util, pero no creo q lo sea para este caso, aceptamos que los isos que grabamos se encuentran en buenas condiciones. Clic en "SIGUIENTE".

Acuerdo de licencia: la licencia expresa de NOVELL. Open Source. Clic en "SIGUIENTE".

Modo de instalacion: elegimos "NUEVA INSTALACION". Clic en "SIGUIENTE".

Reloj y zona horaria: en "ZONA" elegimos "America cenntral...", en Zona horaria "Buenos Aires".

Cambiamos fecha y hora si es necesario haciendo clic en el boton "Cambiar" de "FECHA & HORA".

Cambiamos a "HORA LOCAL" en "RELOJ DE HARDWARE...". Clic en "SIGUIENTE".

Seleccion de escritorio: tenemos 3 opciones: KDE, GNOME u OTRO. Personalmente de entrada me inclino a GNOME que es un GRAN entorno grafico e intuitivo, pero mas tarde probaremos el "SISTEMA GRAFICO MINIMO" que ofrece la opcion "OTRO". elegimos GNOME. Clic en "SIGUIENTE".

Configuracion de la instalacion: probaremos instalar desde cero un unico sist operativo con reiserfs que es por defecto lo que suse 10 ofrece. A partir de ahora comienza la instalacion. Espacio requerido: 2.4 GB de disco.

> Una vez terminada la instalacion, llenamos los campos restantes: contraseña de root y su cifrado lo deje tipo DES.

Configuracion de RED. En la opcion cortafuegos: desabilito el cortafuegos. Esto es importante, ya que el CORTAFUEGOS (FIREWALL) puede denegar puertos que son necesarios para ejecutar ciertos servicios como DHCP o TFTP.

En tarjeta de red cambiamos la configuracion DHCP por estatica con la IP 192.168.0.254 (para

nuestro server).

Cambiamos los nombres en nombre de host por: "server" y en nombre de dominio por "ltsp". Esto se hace porque la instalacion por defecto de LTSP utiliza el nombre "server" para el servidor y el dominio "ltsp" por defecto para su entorno.

Si tenemos DNS de algun ISP lo ponemos en "servidor de nombres n".

En "enrutado" si tenemos alguna pasarela la colocamos aqui.

Todo lo demas que vemos en "configuracion de red" lo dejamos por defecto como esta.
Ahora se guardara la configuracion de la red y se reiniciaran algunos servicios relacionados con la conectividad del equipo.

Probar conexion a Internet. lo pasé por alto.

Metodo de autentificacion de usuarios. Lo dejo por defecto en "local".

Ahora nos pide que ingresemos un "nuevo usuario local", lo cual lo haremos para probar nuestro LTSP mas adelante. Destildamos el "inicio de sesion automatico".

> Notas de la version y configuracion de hardware
Aqui podemos leer lo ultimo que trae opensuse. Con respecto a la deteccion de hardware, fue sin problemas. Detecto la placa de red, de video, etc sin ningun problema.

>Inicio de sesion.
Entramos a la pantalla grafica de login como root.

Bajamos de internet LTSP 4.1.1 de ltsp.org (busquen donde esta en "DOWNLOADS", bajen el .iso o las carpetas), que es el que vamos a usar. lo alojamos en la carpeta root.

Bajamos "ltsp-utils-0.25.0.tar.gz", lo extraemos alli mismo, nos quedara una carpeta "ltsp utils. Ingresamos a este directorio. Abrimos un terminal.

Ingresamos: `./install.sh`

La instalacion del sistema instalador de LTSP 4.1.1 esta lista, pero todavia no va a arrancar, debemos instalar la libreria "libwww-perl". vamos a YAST y lo instalamos (se encuentra en el CD nro. 4).

NOTA: Uso el ltsp-utils 0.25 en vez del 0.1 porque de esta forma me aseguro que en un futuro si quiero actualizar LTSP me sirva este instalador para futuras nuevas versiones.

Corremos las "ltsp-utils" ingresando:

```
# ltspadmin
```

Podemos en este momento chequear los servicios necesarios que hacen funcionar a LTSP, para eso nos vamos a "configurar LTSP".

Chequea el runlevel, si hay placas de red, si esta corriendo dhcp y tftp. Chequea el portmapper (para

nfs) y el server nfs. Luego chequea el login remoto xdmcp y varios archivos de configuracion.

ingresamos con enter y luego presionamos s para ver el estado de los servicios.

vemos que faltan casi todos los servicios ser instalados.

Volvemos atras y configuramos la instalacion de LTSP:

Entramos en configurar opciones del instalador:
Nos pregunta donde tenemos el instalador, como recordamos esta en la carpeta root.
ingresamos: `file:///root/ltsp-4.1.1`

Ahora nos pregunta en donde queremos alojar el arbol de directorios de LTSP, por defectop es /opt/ltsp, presionamos enter.

Si tenemos proxy lo indicamos aqui de la siguiente forma [host]:[puerto]. de no tener proxy, con enter salteamos esta opcion.

Nos pregunta si tenemos ftp proxy, hacemos lo mismo que en el caso anterior.

Confirmamos y vamos a la opcion instalar/actualizar paquetes LTSP.

Vemos ahora una lista de los paquetes a instalar, seguimos las instrucciones.
Procedemos a instalar los paquetes (todos).

Una vez instalados procedemos a configurar manualmente el resto de la implementacion:

Lo primero que vamos a hacer es instalar tftp y dhcp-server. Esto deberemos hacerlo como siempre desde Yast.

Una vez instalado, salimos de ltspadmin y volvemos a entrar, asi actualizamos la ventana de servicios (es la unica forma).

Chequeamos los servicios y ahora podemos ver que estan todos instalados.

> Es hora de dejar los servicios y archivos de configuracion restantes funcionando.

Vamos a configurar los servicios manualmente.

la opcion [1] no tocarla, aunque SuSe viene de debian y el runlevel de debian es 2 aqui es como red hat, 5.

Si tenemos mas de una placa de red entramos a la opcion 2, de todas formas entramos y checamos que este seleccionada la placa que deseamos usar para LTSP.

El topico 3 es referente a la configuracion de dhcp. Podemos ver que dhcp ya esta habilitado, pero debemos habilitarlo para que cada vez q arranque el server (la aplicacion nos guiara para dejarlo seteado en ON al arranque), éste tambien lo haga y de paso tambien, creamos un archivo dhcpd.conf de ejemplo que nos sera tremendamente util mas tarde.
NOTA: Si no queda habilitado dhcp para funcionar desde el arranque, podemos tranquilamente

con Yast habilitarlo en servicios.

La opcion 4 habilita tftp.

La opcion 5 configura portmapper. No hace falta, ya esta configurado.

La opcion 6 configura NFS. entramos y aceptamos que NFS arranque cada vez que bootea el server.

La opcion 7 habilita gdm para recibir llamadas remotas. Lo habilitamos. Luego decimos que no a la desabilitacion de LOGIN grafico, por razones obvias.

Por ultimo creamos los archivos de configuracion necesarios para LTSP, estos son: hosts, hosts.allow, exports y lts.conf.

Ahora en SuSe vamos a buscar los archivos de configuracion de los servicios para modificar pequeños puntos.

DHCP-SERVER
Lo arrancamos, paramos y rearrancamos con: /etc/init.d/dhcpd start-stop-restart.

Apenas instalado nos dara el error siguiente: "DHCPD_INTERFACE en /etc/sysconfig/dhcpd esta vacia!"
Este archivo de configuracion es analogo al de Debian que es /etc/defaults.
Entramos al archivo dhcpd y donde encontramos la variable DHCPD_INTERFACE, agregamos entre las comillas "eth0", guardamos los cambios y rearrancamos dhcp como fue explicado anteriormente.

Ahora el error que nos dara es que no encuentra la red actual, eso es logico porque el archivo dhcpd.conf por defecto apunta a otra red.
Solo tenemos que renombrar este dhcpd.conf por dhcpd.conf.old y el dhcpd.conf.sample que nos instalo la aplicacion ltspadmin por dhcpd.conf.

Reiniciamos nuevamente dhcpd. Debemos tener mensaje de "done".
Ahora entramos a /etc/dhcpd.conf y configuramos un par de cosas:

Una cosa que aprendi es que si hay varios servidores dhcp la cosa se torna dificil, hay colisiones de paquetes, etc.
En la 2da linea de texto incluimos:

```
authoritative;
```

Cambiamos si es necesario las ubicaciones de los DNS asi como tambien la de gateway (routers).
En domain-name cambiamos por ltsp.

Como estamos basados en ETHERBOOT, debemos cambiar el kernel que muestra por defecto que es: vmlinuz-2.6.16.1-ltsp1 por vmlinuz-2.4.26-ltsp3.

TFTP:
Basicamente, reiniciando el demonio xined es suficiente, de todas formas debemos saber que tftp debe apuntar al directorio /tftpboot que es el que creó la instalacion de LTSP y es donde se alojan

los kernels de arranque del entorno en el cliente.
En `/etc/xinetd/tftp` debemos comprobar que este apuntando el servicio a `/tftpboot` corriendo en modo seguro (-s). El modo seguro le dice a tftpd que el path a donde apuntan los ficheros es ABSOLUTO, no relativo.

NFS
Vamos a `/etc/exports` y cambiamos `/opt/ltsp` por `/opt/ltsp/i386` (podemos darnos cuenta que es la misma linea de dhcpd.conf en option-root-path).

Creamos a mano el directorio /var/opt/swap/files que es el directorio donde se alojara la memoria de intercambio de los terminales con poca memoria RAM. NOTA crear primero el sirectorio ltsp y luego el swapfiles en el path /var/opt.

Reiniciamos el servicio:

```
# /etc/init.d/nfsserver restart
```

Y comprobamos los directorios compartidos:

```
# showmount -e
```

Veremos listados los directorios de /etc/exports

Reiniciamos Xinetd:

```
# /etc/init.d/xinetd restart
```

Chequeamos todos los servicios de nuevo con ltspadmin.

Ahora vamos a la configuracion de XDMCP:

Vamos a Aplicaciones > Sistema > Configuracion > Configuracion de pantalla de entrada (Login Screen Setup)

Vamos a la pestaña "general" y habilitamos un tema para la entrada remota, vemos que previamente lo teniamos desabilitado.

Vamos a "seguridad" y comprobamos que XDMCP este habilitado.

En la pestaña "XDMCP" le decimos cuantos clientes vamos a colgar a gdm en "Maximo de sesiones remotas".

Por último, vamos a Yast y desinstalamos el generador de logs del sistema que es "syslog-ng".
Una vez desinstalado, instalamos "syslog" (se encuentra en el CD 1).
Una vez instalado vamos a /etc/sysconfig/syslog y en la linea 20 aprox. encontramos una declaracion de variable:

```
SYSLOGD_PARAMS=""
```

Lo reemplazamos por:

```
SYSLOGD_PARAMS="-r"
```

Esto es para que sea posible la generacion remota de logs. Syslogd de esta forma escucha a la red. Por defecto syslog esta en modo "standalone" o sea local.

REINICIAMOS EL SERVER.
Fin de la implementacion de LTSP en Open SuSe 10.0

NOTAS:
Esta implementacion esta basada en ETHERBOOT, por favor referirse al manual oficial de LTSP para mas informacion o a www.etherboot.org.
Si los clientes tienen menos de 32 MB de RAM se experimentara continuamente freezamientos (congelamiento) de las terminales, especialmente cuando se navegue por Internet, esto se soluciona creando memoria virtual para ese cliente en particular, utilizando el metodo NFS SWAP. Para mas informacion por favor remitase al manual oficial de LTSP.
Sitio Oficial:
www.ltsp.org

Alberto Castillo
Ministerio de Educacion de la Pcia. de Cordoba
Gerencia de Sistemas
Proyecto EduLin (Educacion & Linux)
Implementador y desarrollador de LTSP e hibridos para Distros basadas en RED HAT, DEBIAN, SUSE.
2004-2007
PERMITIDO SU USO TOTAL O PARCIAL. SOLO MENCIONAR LAS FUENTES
14/18 de Mayo de 2007

ANEXO 15 Entrada tipica de lts.conf (configuracion de clientes LTSP)

Las entradas lts.conf residen en:
`/opt/ltsp/i386/etc/lts.conf`
`/opt/ltsp/i386/etc/lts.conf.readme`

Archivo lts.conf de ejemplo:

```
[Default]
        SERVER                    = 192.168.0.254
        XSERVER                   = vesa
        X_MOUSE_PROTOCOL          = "PS/2"
        X_MOUSE_DEVICE            = "/dev/psaux"
        X_MOUSE_RESOLUTION        = 400
        X_MOUSE_BUTTONS           = 3
        USE_XFS                   = N
        SCREEN_01                 = startx
       SCREEN_02                  = shell
        USE_NFS_SWAP              = Y
        SWAPFILE_SIZE             = 96m
        SOUND                     = N
```

Esqueleto de lts.conf
Se divide en secciones independientes:

[default]
.
[ws001]
.
[WS002]
.
[192.168.0.3]
.
[00:50:56:59:7F:81]

NOTA: [ws001] & [WS001] son lo mismo para lts.conf

Dentro de las secciones encontramos las OPCIONES, las cuales nos permiten configurar correctamente nuestros clientes.

*NOTA: ***los manuales oficiales de ltsp 3 & 4 son una enorme ayuda!*** *

OPCIONES de Servidor, memoria virtual, sonido en clientes (mas importantes):

SERVER
Este es el servidor utilizado por XDM_SERVER, TELNET_HOST, XFS_SERVER y SYSLOG_HOST, si alguno de los mencionados no está especificado explícitamente.
Si tienes una máquina que está actuando como el servidor de todo, entonces puedes especificar la dirección IP solamente aquí y omitir el resto. Si ningún valor es elegido, se toma por defecto 192.168.0.254.

NFS_SERVER
Especifica la dirección IP del servidor desde donde el sistema de archivos /home es montado. Por defecto se utiliza la entrada SERVER.

USE_NFS_SWAP
Pon Y aquí si quieres habilitar intercambio sobre NFS. Por defecto es N.

SWAPFILE_SIZE
Aquí es donde puedes controlar el tamaño del archivo de intercambio. El tamaño por defecto es 64m.

SOUND
Necesitas poner Y en esta entrada y ejecutar el script rc.sound para configurar la placa de sonido y el demonio. Por defecto es N.

OPCIONES DE X (mas importantes):

SCREEN_01 through SCREEN_12
Hasta 12 scripts de pantalla pueden ser especificados para un cliente. Esto te dara hasta 12 inicios de sesion por maquina, cada una accesible presionando Ctrl+Alt+F1 hasta Ctrl−Alt−F12.

```
SCREEN_01      = startx
SCREEN_02      = shell
```

Actualmente, los posibles valores son:

- ➔ `startx`
- ➔ `telnet`
- ➔ `rdesktop`
- ➔ `shell`

XSERVER
Esta entrada define cuál servidor X la estación de trabajo correrá. Para placas PCI y AGP, este parámetro no debería ser necesario. El script rc.local debería poder autodetectar la tarjeta. Puedes también poner auto aquí a fin de indicar que se tratará de autodetectar el video.
Para placas ISA, o para especificar un servidor X en particular, puedes poner el nombre del driver o servidor X aquí.
Si el valor comienza con 'XF86_', entonces XFree 3.3.6 será utilizado. De otro modo, XFree 4.1.x lo será. El valor por defecto es auto.
Ver mas info en ltsp.org

Valores permitidos para Xorg:

El Xserver Xorg es un Servidor X modular que tiene un kernel generico y muchos modulos cargables.
Esta es la actual lista de modulos:

* apm
* ark
* ati
* atimisc
* chips
* cirrus
* cyrix
* dummy
* fbdev
* geode
* glint
* i128
* i740
* i810
* mga
* neomagic
* nsc
* nv
* r128
* radeon
* rendition
* s3
* s3virge
* savage
* siliconmotion
* sis
* tdfx
* tga
* trident
* tseng
* **vesa # infalible**
* vga
* via
* vmware

X_MODE_0 through X_MODE_2

Hasta 3 Modelines o resoluciones pueden ser configuradas para la terminal. Esta entrada puede tomar dos tipos diferentes de valores. Puede ser tanto una resolución como un modeline completo:

```
X_MODE_0 = 800x600
```

o bien

```
X_MODE_0 = 800x600 60.75 800 864 928 1088 600 616 621 657 –HSync -Vsync
```

Si ninguna de las entradas X_MODE_x es especificada, entonces se utilizarán los modelines por defecto, y las resoluciones serán de 1024x768, 800x600 y 640x480.
Si una o mas entradas X_MODE_x es especificada, entonces no se tendrán en consideración los modelines por defecto.

OPCIONES de raton (mas importantes):

X_MOUSE_PROTOCOL
Cualquier valor que funcione con el Protocolo de Puntero XFree86 puede ser puesto aquí. Los valores típicos incluyen 'Microsoft' y 'PS/2'. El valor por defecto es 'PS/2'.
El valor 'Microsoft' corresponde a los tipicos Mouses seriales con fichas DB-9.

*NOTA: *** No todos los Mouses seriales funcionan en el entorno LTSP, por ejemplo, los ratones marca "NogaNet" (Argentina) poseen algun tipo de incompatibilidad con la lectura de puertos serie del Kernel, asi que el resultado es simplemente que no funcionan.****

X_MOUSE_DEVICE
Este es el dispositivo al cual el mouse está conectado. Si es un mouse serial, debería ser un puerto serie, como /dev/ttyS0 o /dev/ttyS1. Si es un mouse PS/2, este valor debería ser /dev/psaux. El valor por defecto es /dev/psaux.

X_MOUSE_RESOLUTION
Este es el valor 'Resolution' que encuentras en cualquier archivo XF86Config. Un valor típico para un mouse serial es 50 y para uno PS/2 es 400. El valor por defecto es 400.

X_BUTTONS
Le dice la sistema cuántos botones tiene el mouse. Usualmente son 2 o 3. El valor por defecto es 3.

X_MOUSE_EMULATE3BTN
Esto le dice al servidor X que debe emular el tercer botón del mouse cuando se presionen simultáneamente los botones izquierdo y derecho. El valor por defecto es N.

X_MOUSE_BAUD
Para ratones seriales, define la tasa de baudios. El valor por defecto es 1200.

OPCIONES de pantalla (mas importantes):

X_COLOR_DEPTH
Este es el número de bits a utilizar para la profundidad del color. Los valores posibles son 8, 15, 16,24 y 32. 8 bits te dará 256 colores, 16 te dará 65536, 24 te dará 16 millones y 32 te dará 4,2 billones de colores. No todos los servidores X soportan estos valores. El valor por defecto es 16.

X_HORZSYNC
Setea la sincronización horizontal del monitor. El valor por defecto es "31−62".

X_VERTREFRESH
Setea el refresco vertical del monitor. Por defecto es "55−90".

Opciones de teclado (mas importantes):

`XkbTypes`
El valor por defecto aquí es la palabra 'default'.

`XkbCompat`
El valor por defecto aquí es la palabra 'default'.

`XkbSymbols`
El valor por defecto aquí es la palabra 'default'.

`XkbModel`
El valor por defecto es 'pc101'.

`XkbLayout`
El valor por defecto es 'us'.

Printer configuration parameters (mas importantes):
Hasta tres (3) impresoras pueden conectarse a un cliente LTSP. Combinaciones de impresoras seriales y paralelas poueden ser configuradas con algunas de las sigueientes entradas en nuestro archivo lts.conf:

`PRINTER_0_DEVICE`
El nombre de dispositivo de la primer impresora.
Nombres tales como `/dev/usb/lp0, /dev/lp0, /dev/ttyS0, or /dev/ttyS1` son permitidos.

`PRINTER_0_TYPE`
El tipo de impresora. Las elecciones válidas son `'U'` para USB, `'P'` para Paralelo y `'S'` para serial.

`PRINTER_0_WRITE_ONLY`
Algunas impresoras detectaran si el puerto es bidireccional (lectura/escritura, no solo escritura) y esperaran ser habilitadas para enviar la informacion, quedandose colgadas en un loop infinito. Seteando el puerto a WRITE-ONLY logramos que la impresora no detecte el puerto. Si no se especifica nada, la opcion 'N' sera usada.

ANEXO 16 dhcpd.conf tipico para UTC/LTSP

```
# Sample configuration file for ISCD dhcpd
#
# Don't forget to set run_dhcpd=1 in /etc/init.d/dhcpd
# once you adjusted this file and copied it to /etc/dhcpd.conf.
#

default-lease-time            21600;
max-lease-time                21600;
ddns-update-style none;
allow booting;
allow bootp;

option subnet-mask            255.255.255.0;
option broadcast-address      192.168.0.255;
option routers                192.168.0.254;
option domain-name-servers    192.168.0.254;
next-server                   192.168.0.254;
option domain-name            "";
option root-path              "192.168.0.254:/opt/edulin";
option option-128 code 128 = string;
option option-129 code 129 = text;
option option-221 code 221 = text;

shared-network WORKSTATIONS {
  subnet 192.168.0.0 netmask 255.255.255.0 {
     range dynamic-bootp 192.168.0.200 192.168.0.253;
     use-host-decl-names       on;
     option log-servers        192.168.0.254;

     # trick from Peter Rundle <peter.rundle@au.interpath.net>
     # newer Macs
     if substring (option vendor-class-identifier, 0, 9) = "AAPLBSDPC"
     {
        filename      "yaboot";
        option vendor-class-identifier "AAPLBSDPC";
     }
     # really old iMacs
     elsif substring (option option-221, 0, 5) = "Apple"
     {
        filename      "yaboot";
        option vendor-class-identifier "AAPLBSDPC";
     }
     # Intel PXE
     elsif substring (option vendor-class-identifier, 0, 9) = "PXEClient"
     {
        # NOTE: kernels are specified in /tftpboot/lts/pxe/pxelinux.cfg/
        filename      "/lts/pxe/pxelinux.0";
     }
     # default to an i386 BOOTP image
     else
     {
        filename      "/lts/vmlinuz";
     }

     if substring (option vendor-class-identifier, 20, 3) = "ppc" {
        option root-path "192.168.0.254:/opt/ltsp/ppc";
     } else {
        option root-path "192.168.0.254:/opt/ltsp/i386";
```

```
     }
  }
}

# example configurations for specifying specific kernels to specific clients
group {
    use-host-decl-names       on;
    option log-servers        192.168.0.254;

    host ws001 {
        hardware ethernet     00:E0:06:E8:00:84;
        fixed-address         192.168.0.1;
        filename              "/lts/vmlinuz.ltsp";
        option option-128 e4:45:74:68:00:00;
        option option-129 "NIC=3c509";
    }
    host ws002 {
        hardware ethernet     00:D0:09:30:6A:1C;
        fixed-address         192.168.0.2;
        filename              "/lts/vmlinuz.ltsp";
        option option-128 e4:45:74:68:00:00;
        option option-129 "NIC=ne";
    }
    host ws003 {
        hardware ethernet     00:D0:09:30:28:B2;
        fixed-address         192.168.0.3;
        # kernels are specified in /tftpboot/lts/boot/pxe/pxelinux.cfg/
        filename              "/lts/boot/pxe/pxelinux.0";
    }

#################################################################
### ¡¡¡ ATENCION: configuracion de ejemplo para cliente UTC    ###
###     Agregar esto a tu configuracion dhcpd existente !!!    ###
#################################################################
host ws101 {
            hardware ethernet 00:50:fc:06:32:3c;
            fixed-address           192.168.0.101;
            filename          "/utc/kernel";
            option option-128 e4:45:74:68:00:00;
            option option-129 "ip=dhcp nfsroot=192.168.0.254:/opt/utc";
            }

#################################################################
###                  ¡¡¡FIN DEL EJEMPLO!!!                     ###
###                  ¡¡¡FIN DEL EJEMPLO!!!                     ###
#################################################################

# Apple Specific Settings
#    host ws007 {
#       hardware ethernet     00:30:65:69:23:60;
#       fixed-address         192.168.0.4;
#       option root-path      "192.168.0.254:/opt/ltsp/ppc";
#       filename              "yaboot";
#       option vendor-class-identifier "AAPLBSDPC";
#    }
}
```

ANEXO 17 /etc/exports tipico para UTC/LTSP

```
## LTS-begin ##
#
# The lines between the 'LTS-begin' and the 'LTS-end' were added
# on: Wed Nov 28 11:19:42 PST 2001 by the ltsp installation script.
# For more information, visit the ltsp homepage
# at http://www.ltsp.org
#
#################### ATENCION ######################
# AGREGAR LA SIGUIENTE LINEA A TU ARCHIVO exports #
# no te olvides de reiniciar servicio             #
####################################################

/opt/utc            192.168.0.0/255.255.255.0(ro,no_root_squash,async)

###################### FIN #######################

/opt/ltsp/i386            192.168.0.0/255.255.255.0(ro,no_root_squash,sync)
/var/opt/ltsp/swapfiles   192.168.0.0/255.255.255.0(rw,no_root_squash,async)

#
# The following entries need to be uncommented if you want
# Local App support in ltsp
#
#/home                    192.168.0.0/255.255.255.0(rw,root_squash,sync)

## LTS-end ##

## LTS-begin ##
#
# The lines between the 'LTS-begin' and the 'LTS-end' were added
# on: Wed Nov 28 11:19:42 PST 2001 by the ltsp installation script.
# For more information, visit the ltsp homepage
# at http://www.ltsp.org
#

/usr/X11R6/lib/X11/fonts   192.168.0.0/255.255.255.0(ro,no_root_squash,sync)
/usr/share/AbiSuite/fonts  192.168.0.0/255.255.255.0(ro,no_root_squash,sync)
/usr/share/fonts           192.168.0.0/255.255.255.0(ro,no_root_squash,sync)
/usr/lib/openoffice/share/fonts
192.168.0.0/255.255.255.0(ro,no_root_squash,sync)

## LTS-end ##

## LTS-begin ##
#
# The lines between the 'LTS-begin' and the 'LTS-end' were added
# on: Wed Nov 28 11:19:42 PST 2001 by the ltsp installation script.
# For more information, visit the ltsp homepage
# at http://www.ltsp.org
#

/opt/ltsp/ppc            192.168.0.0/255.255.255.0(ro,no_root_squash,sync)

## LTS-end ##
```

ANEXO 18 Como usar los clientes LTSP como pequeños servidores de impresion (Configuración lado cliente)

Impresoras en clientes finos:
LTSP puede soportar hasta 3 impresoras en un cliente fino (2 seriales y 1 paralela)
Actualmente, las impresoras USB tambien son soportadas.

Ejemplos mas comunes:

<Configuracion lado cliente - entradas en lts.conf:>
Serial printer

```
[ws001]
   PRINTER_0_TYPE     = S
   PRINTER_0_DEVICE   = /dev/ttyS0
   PRINTER_0_SPEED    = 9600
```

Parallel printer

```
[ws002]
   PRINTER_0_TYPE     = P
   PRINTER_0_DEVICE   = /dev/lp0
   PRINTER_0_WRITE_ONLY = Y
```

USB Printer

```
[ws001]
   PRINTER_0_TYPE     = U
   PRINTER_0_DEVICE   = /dev/usb/lp0
```

Cargar previamente el modulo PRINTER_USB al principio de lts.conf:

```
MODULE_01          = usblp
```

ANEXO 19 Como usar los clientes LTSP como pequeños servidores de impresion (Configuración lado Servidor)

Primero debemos hacer un par de cosas en nuestro servidor, antes
Configurar una impresora del lado del servidor es cuestión de definir una cola de impresión, utilizando la herramienta de configuración apropiada en el servidor.
Una vez que has lanzado la herramienta de configuración, debes agregar una nueva impresora.
El programa **lp_server** le permite a la estación de trabajo emular un servidor de impresión *HP JetDirect*, por lo que sólo necesitas crear una impresora JetDirect.

Necesitas luego darle a la cola un nombre. Este nombre puede ser cualquier cosa (trata de que sea un nombre significativo) y debe contener sólo los siguientes caracteres:

- *"a−z"* *letras en minúsculas*
- *"A−Z"* *letras en mayúsculas*
- *"0−9"* *dígitos numéricos*
- *"−"* *guión medio*
- *"_"* *guión bajo*

El nombre seleccionado en el ejemplo de arriba es `ws001_lp`. Este nombre hace que sea fácil saber que la impresora está asociada con `ws001`.

Hay dos campos necesarios para comunicarse con la impresora:

1. **La dirección IP** y el nombre de host de la estación de trabajo a la cual la impresora está asociada.
2. **El puerto TCP** en el cual el demonio lp_server está corriendo.

La primera impresora que conectes a la estación de trabajo estará en el puerto `TCP/IP 9100`. La segunda lo hará en el puerto `TCP/IP 9101` y la tercera en el `TCP/IP 9102`.

ANEXO 20 LOCAL MEDIA

"Introduction:
LDA(Local Drive Access) allows LTSP users to read or write data to a floppy or read data from a CD-ROM. Currently LDA supports CD-ROMs, floppies, DVDs, and USB memory sticks in read/write mode only."

Teoria de funcionamiento basico:
Los dispositivos se montan con *supermount*, de esta forma se asegura de que solo se ponga en marcha cuando se le haga una peticion al dispositivo (la puerta del CD se bloquea mientras es usado). El terminal genera una configuracion samba y comparte los dispositivos, asi de esta forma pueden ser accedidos desde cualquir lugar.

Escribiendo
`$ smb://nobody@<IP_OF_TERMINAL>/drives` podemos acceder a los dispositivos del terminal.

Configuracion:
Server
Instalar el paquete `samba-client` (en debian `apt-get install smbfs`, en RH `yum install samba-client`).

Instalar los scripts (buscarlos en www.ltsp.org):

```
LDA-setup.sh
LDA-shutdown.sh
```

Y ubicarlos en `/usr/local/bin`

Si usa kdm:
Escriba '`/usr/local/bin/LDA-setup.sh`' en la linea 3 de `/etc/kde/kdm/Xstartup`
Escriba '`/usr/local/bin/LDA-shutdown.sh`' en la linea 3 de `/etc/kde/kdm/Xreset`

Si usa gdm:
Escriba '`/usr/local/bin/LDA-setup.sh`' en linea 3 de `/etc/X11/gdm/PostLogin/Default`
Escriba '`/usr/local/bin/LDA-shutdown.sh`' linea 3 de `/etc/X11/gdm/PostSession/Default`

En los clientes:
Entradas en lts.conf

"LDA has two options: LOCAL_DEVICE_XX = /dev/<device>:<devicename>"

Ejemplo:
```
LOCAL_DEVICE_01 = /dev/hdc:cdrom
LOCAL_DEVICE_02 = /dev/fd0:floppy
```

NOTA: Si no funcionan los CD-ROMs, agregar el siguiente modulo:
`MODULE_01 = ide_generic`
Dispositivos USB solo agregar:

```
HOTPLUG = Y
```

NOTA: si no funcionan los dispositivos USB, editar:

```
$LTSP_DIR/i386/etc/rc.usb
```

agregar '`usb-storage`' despues del modulo '`usbkbd`'

DIAGNOSTICO:
Levantemos en modo consola la terminal o agreguemos una en lts.conf:

```
SCREEN_02 = shell
```

Reiniciemos la terminal e ingresemos al shell, ejecutemos:

```
# mount | grep supermount
```

Mostrara los dispositivos asociados a la terminal.
Ahora insertamos un diskette y lo listamos:

```
# ls /tmp/drives/floppy
```

Por ultimo desde el server y en modo grafico con el Nautilus o Konkeror escribimos:

```
smb://nobody@<IP_terminal>
```

Deberias ver los dispositivos montados con supermount.
Para mas info: ltsp.org

ANEXO 21 Mapa de teclado Ingles para svncviewer

Contenido en las fuentes, archivo keys.h

```
  /*
    * Definition of the standard US keyboard -- 27JUL99wzk
    */


#define     KT_NORMAL   0
#define     KT_SHIFTED  1
#define     KT_ALTGR    2
#define     KT_CONTROL  3


int keytable[NR_KEYS][4] = {

  /* 000 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
  /* 001 */ XK_Escape,  XK_Escape,  NoSymbol,   NoSymbol,
  /* 002 */ XK_1,       XK_exclam,  NoSymbol,   NoSymbol,
  /* 003 */ XK_2,       XK_at,            NoSymbol,   NoSymbol,
  /* 004 */ XK_3,       XK_numbersign,    NoSymbol,   NoSymbol,
  /* 005 */ XK_4,       XK_dollar,  NoSymbol,   NoSymbol,
  /* 006 */ XK_5,       XK_percent, NoSymbol,   NoSymbol,
  /* 007 */ XK_6,       XK_asciicircum,   NoSymbol,   NoSymbol,
  /* 008 */ XK_7,       XK_ampersand,     NoSymbol,   NoSymbol,
  /* 009 */ XK_8,       XK_asterisk,      NoSymbol,   NoSymbol,
  /* 010 */ XK_9,       XK_parenleft,     NoSymbol,   NoSymbol,
  /* 011 */ XK_0,       XK_parenright,    NoSymbol,   NoSymbol,
  /* 012 */ XK_minus,   XK_underscore,    NoSymbol,   NoSymbol,
  /* 013 */ XK_equal,   XK_plus,    NoSymbol,   NoSymbol,
  /* 014 */ XK_BackSpace,     XK_BackSpace,     NoSymbol,   NoSymbol,
  /* 015 */ XK_Tab,           XK_ISO_Left_Tab, NoSymbol,    NoSymbol,
  /* 016 */ XK_q,       XK_Q,       NoSymbol,   NoSymbol,
  /* 017 */ XK_w,       XK_W,       NoSymbol,   NoSymbol,
  /* 018 */ XK_e,       XK_E,       NoSymbol,   NoSymbol,
  /* 019 */ XK_r,       XK_R,       NoSymbol,   NoSymbol,
  /* 020 */ XK_t,       XK_T,       NoSymbol,   NoSymbol,
  /* 021 */ XK_y,       XK_Y,       NoSymbol,   NoSymbol,
  /* 022 */ XK_u,       XK_U,       NoSymbol,   NoSymbol,
  /* 023 */ XK_i,       XK_I,       NoSymbol,   NoSymbol,
  /* 024 */ XK_o,       XK_O,       NoSymbol,   NoSymbol,
  /* 025 */ XK_p,       XK_P,       NoSymbol,   NoSymbol,
  /* 026 */ XK_bracketleft,   XK_braceleft,     NoSymbol,   NoSymbol,
  /* 027 */ XK_bracketright, XK_braceright,     NoSymbol,   NoSymbol,
  /* 028 */ XK_Return,  XK_Return,  NoSymbol,   NoSymbol,
  /* 029 */ XK_Control_L,     XK_Control_L,     NoSymbol,   NoSymbol,
  /* 030 */ XK_a,       XK_A,       NoSymbol,   NoSymbol,
  /* 031 */ XK_s,       XK_S,       NoSymbol,   NoSymbol,
  /* 032 */ XK_d,       XK_D,       NoSymbol,   NoSymbol,
  /* 033 */ XK_f,       XK_F,       NoSymbol,   NoSymbol,
  /* 034 */ XK_g,       XK_G,       NoSymbol,   NoSymbol,
  /* 035 */ XK_h,       XK_H,       NoSymbol,   NoSymbol,
  /* 036 */ XK_j,       XK_J,       NoSymbol,   NoSymbol,
  /* 037 */ XK_k,       XK_K,       NoSymbol,   NoSymbol,
  /* 038 */ XK_l,       XK_L,       NoSymbol,   NoSymbol,
  /* 039 */ XK_semicolon,     XK_colon,   NoSymbol,   NoSymbol,
  /* 040 */ XK_apostrophe,    XK_quotedbl,      NoSymbol,   NoSymbol,
  /* 041 */ XK_grave,   XK_asciitilde,    NoSymbol,   NoSymbol,
  /* 042 */ XK_Shift_L, XK_Shift_L, NoSymbol,   NoSymbol,
```

```
/* 043 */ XK_backslash,     XK_bar,           NoSymbol,   NoSymbol,
/* 044 */ XK_z,       XK_Z,       NoSymbol,   NoSymbol,
/* 045 */ XK_x,       XK_X,       NoSymbol,   NoSymbol,
/* 046 */ XK_c,       XK_C,       NoSymbol,   NoSymbol,
/* 047 */ XK_v,       XK_V,       NoSymbol,   NoSymbol,
/* 048 */ XK_b,       XK_B,       NoSymbol,   NoSymbol,
/* 049 */ XK_n,       XK_N,       NoSymbol,   NoSymbol,
/* 050 */ XK_m,       XK_M,       NoSymbol,   NoSymbol,
/* 051 */ XK_comma,   XK_less,    NoSymbol,   NoSymbol,
/* 052 */ XK_period,  XK_greater, NoSymbol,   NoSymbol,
/* 053 */ XK_slash,   XK_question,      NoSymbol,   NoSymbol,
/* 054 */ XK_Shift_R, XK_Shift_R, NoSymbol,   NoSymbol,
/* 055 */ XK_KP_Multiply,   XK_KP_Multiply,   NoSymbol,   NoSymbol,
/* 056 */ XK_Alt_L,   XK_Meta_L,  NoSymbol,   NoSymbol,
/* 057 */ XK_space,   XK_space,   NoSymbol,   NoSymbol,
/* 058 */ XK_Caps_Lock,     XK_Caps_Lock,     NoSymbol,   NoSymbol,
/* 059 */ XK_F1,            XK_F1,            NoSymbol,   NoSymbol,
/* 060 */ XK_F2,            XK_F2,            NoSymbol,   NoSymbol,
/* 061 */ XK_F3,            XK_F3,            NoSymbol,   NoSymbol,
/* 062 */ XK_F4,            XK_F4,            NoSymbol,   NoSymbol,
/* 063 */ XK_F5,            XK_F5,            NoSymbol,   NoSymbol,
/* 064 */ XK_F6,            XK_F6,            NoSymbol,   NoSymbol,
/* 065 */ XK_F7,            XK_F7,            NoSymbol,   NoSymbol,
/* 066 */ XK_F8,            XK_F8,            NoSymbol,   NoSymbol,
/* 067 */ XK_F9,            XK_F9,            NoSymbol,   NoSymbol,
/* 068 */ XK_F10,           XK_F10,           NoSymbol,   NoSymbol,
/* 069 */ XK_Num_Lock,      XK_Num_Lock,      NoSymbol,   NoSymbol,
/* 070 */ XK_Scroll_Lock,   XK_Scroll_Lock,   NoSymbol,   NoSymbol,
/* 071 */ XK_KP_Home, XK_KP_7,    NoSymbol,   NoSymbol,
/* 072 */ XK_KP_Up,   XK_KP_8,    NoSymbol,   NoSymbol,
/* 073 */ XK_KP_Prior,      XK_KP_9,    NoSymbol,   NoSymbol,
/* 074 */ XK_KP_Subtract,   XK_KP_Subtract,   NoSymbol,   NoSymbol,
/* 075 */ XK_KP_Left, XK_KP_4,    NoSymbol,   NoSymbol,
/* 076 */ XK_KP_Begin,      XK_KP_5,    NoSymbol,   NoSymbol,
/* 077 */ XK_KP_Right,      XK_KP_6,    NoSymbol,   NoSymbol,
/* 078 */ XK_KP_Add,  XK_KP_Add,  NoSymbol,   NoSymbol,
/* 079 */ XK_KP_End,  XK_KP_1,    NoSymbol,   NoSymbol,
/* 080 */ XK_KP_Down, XK_KP_2,    NoSymbol,   NoSymbol,
/* 081 */ XK_KP_Next, XK_KP_3,    NoSymbol,   NoSymbol,
/* 082 */ XK_KP_Insert,     XK_KP_0,    NoSymbol,   NoSymbol,
/* 083 */ XK_KP_Delete,     XK_KP_Decimal,    NoSymbol,   NoSymbol,
/* 084 */ XK_Sys_Req, NoSymbol,   NoSymbol,   NoSymbol,
/* 085 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 086 */ XK_less,    XK_greater, NoSymbol,   NoSymbol,
/* 087 */ XK_F11,           XK_F11,           NoSymbol,   NoSymbol,
/* 088 */ XK_F12,           XK_F12,           NoSymbol,   NoSymbol,
/* 089 */ XK_Home,    XK_Home,    NoSymbol,   NoSymbol,
/* 090 */ XK_Up,            XK_Up,            NoSymbol,   NoSymbol,
/* 091 */ XK_Prior,   XK_Prior,   NoSymbol,   NoSymbol,
/* 092 */ XK_Left,    XK_Left,    NoSymbol,   NoSymbol,
/* 093 */ XK_Begin,   XK_Begin,   NoSymbol,   NoSymbol,
/* 094 */ XK_Right,   XK_Right,   NoSymbol,   NoSymbol,
/* 095 */ XK_End,           XK_End,           NoSymbol,   NoSymbol,
/* 096 */ XK_KP_Enter,      XK_KP_Enter,      NoSymbol,   NoSymbol,
/* 097 */ XK_Control_R,     XK_Control_R,     NoSymbol,   NoSymbol,
/* 098 */ XK_KP_Divide,     XK_KP_Divide,     NoSymbol,   NoSymbol,
/* 099 */ XK_Print,   XK_Print,   NoSymbol,   NoSymbol,
/* 100 */ XK_Alt_R,   XK_Alt_R,   NoSymbol,   NoSymbol,
/* 101 */ XK_Control_R,     XK_Control_R,     NoSymbol,   NoSymbol,
/* 102 */ XK_Home,    XK_Home,    NoSymbol,   NoSymbol,
/* 103 */ XK_Up,            XK_Up,            NoSymbol,   NoSymbol,
```

```
/* 104 */ XK_Prior,   XK_Prior,   NoSymbol,   NoSymbol,
/* 105 */ XK_Left,    XK_Left,    NoSymbol,   NoSymbol,
/* 106 */ XK_Right,   XK_Right,   NoSymbol,   NoSymbol,
/* 107 */ XK_End,           XK_End,           NoSymbol,   NoSymbol,
/* 108 */ XK_Down,    XK_Down,    NoSymbol,   NoSymbol,
/* 109 */ XK_Next,    XK_Next,    NoSymbol,   NoSymbol,
/* 110 */ XK_Insert,  XK_Insert,  NoSymbol,   NoSymbol,
/* 111 */ XK_Delete,  XK_Delete,  NoSymbol,   NoSymbol,
/* 112 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 113 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 114 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 115 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 116 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 117 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 118 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 119 */ XK_Pause,   XK_Pause,   NoSymbol,   NoSymbol,
/* 120 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 121 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 122 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 123 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 124 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 125 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 126 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol,
/* 127 */ NoSymbol,   NoSymbol,   NoSymbol,   NoSymbol

    };
```

ANEXO 21b Mapa de teclado Español para svncviewer

Contenido en las fuentes, reemplaza al archivo keys.h antes de la compilación. (hackeado por Pablo Perez de Angelis y Alberto Castillo).

```
keynorm[0] = NoSymbol;
keynorm[1] = XK_Escape;
keynorm[2] = XK_1;
keynorm[3] = XK_2;
keynorm[4] = XK_3;
keynorm[5] = XK_4;
keynorm[6] = XK_5;
keynorm[7] = XK_6;
keynorm[8] = XK_7;
keynorm[9] = XK_8;
keynorm[10] = XK_9;
keynorm[11] = XK_0;
keynorm[12] = XK_apostrophe;
keynorm[13] = XK_questiondown;
keynorm[14] = XK_Delete;
keynorm[15] = XK_Tab;
keynorm[16] = XK_q;
keynorm[17] = XK_w;
keynorm[18] = XK_e;
keynorm[19] = XK_r;
keynorm[20] = XK_t;
keynorm[21] = XK_y;
keynorm[22] = XK_u;
keynorm[23] = XK_i;
keynorm[24] = XK_o;
keynorm[25] = XK_p;
keynorm[26] = XK_dead_grave;
keynorm[27] = XK_plus;
keynorm[28] = XK_Return;
keynorm[29] = XK_Control_L;
keynorm[30] = XK_a;
keynorm[31] = XK_s;
keynorm[32] = XK_d;
keynorm[33] = XK_f;
keynorm[34] = XK_g;
keynorm[35] = XK_h;
keynorm[36] = XK_j;
keynorm[37] = XK_k;
keynorm[38] = XK_l;
keynorm[39] = XK_ntilde;
keynorm[40] = XK_braceleft;
keynorm[41] = XK_at;
keynorm[42] = XK_Shift_L;
keynorm[43] = XK_braceright;
keynorm[44] = XK_z;
keynorm[45] = XK_x;
keynorm[46] = XK_c;
keynorm[47] = XK_v;
keynorm[48] = XK_b;
keynorm[49] = XK_n;
keynorm[50] = XK_m;
keynorm[51] = XK_comma;
```

```
keyshift[0] = NoSymbol;
keyshift[1] = XK_Escape;
keyshift[2] = XK_exclam;
keyshift[3] = XK_quotedbl;
keyshift[4] = XK_numbersign;
keyshift[5] = XK_dollar;
keyshift[6] = XK_percent;
keyshift[7] = XK_ampersand;
keyshift[8] = XK_slash;
keyshift[9] = XK_parenleft;
keyshift[10] = XK_parenright;
keyshift[11] = XK_equal;
keyshift[12] = XK_question;
keyshift[13] = XK_exclamdown;
keyshift[14] = XK_BackSpace;
keyshift[15] = XK_ISO_Left_Tab;
keyshift[16] = XK_Q;
keyshift[17] = XK_W;
keyshift[18] = XK_E;
keyshift[19] = XK_R;
keyshift[20] = XK_T;
keyshift[21] = XK_Y;
keyshift[22] = XK_U;
keyshift[23] = XK_I;
keyshift[24] = XK_O;
keyshift[25] = XK_P;
keyshift[26] = XK_dead_circumflex;
keyshift[27] = XK_asterisk;
keyshift[28] = XK_Return;
keyshift[29] = XK_Control_L;
keyshift[30] = XK_A;
keyshift[31] = XK_S;
keyshift[32] = XK_D;
keyshift[33] = XK_F;
keyshift[34] = XK_G;
keyshift[35] = XK_H;
keyshift[36] = XK_J;
keyshift[37] = XK_K;
keyshift[38] = XK_L;
keyshift[39] = XK_Ntilde;
keyshift[40] = XK_bracketleft;
keyshift[41] = XK_at;
keyshift[42] = XK_Shift_L;
keyshift[43] = XK_bracketright;
keyshift[44] = XK_Z;
keyshift[45] = XK_X;
keyshift[46] = XK_C;
keyshift[47] = XK_V;
keyshift[48] = XK_B;
keyshift[49] = XK_N;
keyshift[50] = XK_M;
keyshift[51] = XK_semicolon;
```

```
keynorm[52] = XK_period;
keynorm[53] = XK_minus;
keynorm[54] = XK_Shift_R;
keynorm[55] = XK_KP_Multiply;
keynorm[56] = XK_Alt_L;
keynorm[57] = XK_space;
keynorm[58] = XK_Caps_Lock;
keynorm[59] = XK_F1;
keynorm[60] = XK_F2;
keynorm[61] = XK_F3;
keynorm[62] = XK_F4;
keynorm[63] = XK_F5;
keynorm[64] = XK_F6;
keynorm[65] = XK_F7;
keynorm[66] = XK_F8;
keynorm[67] = XK_F9;
keynorm[68] = XK_F10;
keynorm[69] = XK_Num_Lock;
keynorm[70] = XK_Scroll_Lock;
keynorm[71] = XK_KP_Home;
keynorm[72] = XK_KP_Up;
keynorm[73] = XK_KP_Prior;
keynorm[74] = XK_KP_Subtract;
keynorm[75] = XK_KP_Left;
keynorm[76] = XK_KP_Begin;
keynorm[77] = XK_KP_Right;
keynorm[78] = XK_KP_Add;
keynorm[79] = XK_KP_End;
keynorm[80] = XK_KP_Down;
keynorm[81] = XK_KP_Next;
keynorm[82] = XK_KP_Insert;
keynorm[83] = XK_KP_Delete;
keynorm[84] = XK_Sys_Req;
keynorm[85] = NoSymbol;
keynorm[86] = XK_less;
keynorm[87] = XK_F11;
keynorm[88] = XK_F12;
keynorm[89] = XK_Home;
keynorm[90] = XK_Up;
keynorm[91] = XK_Prior;
keynorm[92] = XK_Left;
keynorm[93] = XK_Begin;
keynorm[94] = XK_Right;
keynorm[95] = XK_End;
keynorm[96] = XK_KP_Enter;
keynorm[97] = XK_Control_R;
keynorm[98] = XK_KP_Divide;
keynorm[99] = XK_Print;
keynorm[100] = XK_Alt_R;
keynorm[101] = XK_Control_R;
keynorm[102] = XK_Home;
keynorm[103] = XK_Up;
keynorm[104] = XK_Prior;
keynorm[105] = XK_Left;
keynorm[106] = XK_Right;
keynorm[107] = XK_End;
keynorm[108] = XK_Down;
keynorm[109] = XK_Next;
keynorm[110] = XK_Insert;
keynorm[111] = XK_Delete;
keynorm[112] = NoSymbol;
```

```
keyshift[52] = XK_colon;
keyshift[53] = XK_underscore;
keyshift[54] = XK_Shift_R;
keyshift[55] = XK_KP_Multiply;
keyshift[56] = XK_Meta_L;
keyshift[57] = XK_space;
keyshift[58] = XK_Caps_Lock;
keyshift[59] = XK_F1;
keyshift[60] = XK_F2;
keyshift[61] = XK_F3;
keyshift[62] = XK_F4;
keyshift[63] = XK_F5;
keyshift[64] = XK_F6;
keyshift[65] = XK_F7;
keyshift[66] = XK_F8;
keyshift[67] = XK_F9;
keyshift[68] = XK_F10;
keyshift[69] = XK_Num_Lock;
keyshift[70] = XK_Scroll_Lock;
keyshift[71] = XK_KP_7;
keyshift[72] = XK_KP_8;
keyshift[73] = XK_KP_9;
keyshift[74] = XK_KP_Subtract;
keyshift[75] = XK_KP_4;
keyshift[76] = XK_KP_5;
keyshift[77] = XK_KP_6;
keyshift[78] = XK_KP_Add;
keyshift[79] = XK_KP_1;
keyshift[80] = XK_KP_2;
keyshift[81] = XK_KP_3;
keyshift[82] = XK_KP_0;
keyshift[83] = XK_KP_Decimal;
keyshift[84] = NoSymbol;
keyshift[85] = NoSymbol;
keyshift[86] = XK_greater;
keyshift[87] = XK_F11;
keyshift[88] = XK_F12;
keyshift[89] = XK_Home;
keyshift[90] = XK_Up;
keyshift[91] = XK_Prior;
keyshift[92] = XK_Left;
keyshift[93] = XK_Begin;
keyshift[94] = XK_Right;
keyshift[95] = XK_End;
keyshift[96] = XK_KP_Enter;
keyshift[97] = XK_Control_R;
keyshift[98] = XK_KP_Divide;
keyshift[99] = XK_Print;
keyshift[100] = XK_Alt_R;
keyshift[101] = XK_Control_R;
keyshift[102] = XK_Home;
keyshift[103] = XK_Up;
keyshift[104] = XK_Prior;
keyshift[105] = XK_Left;
keyshift[106] = XK_Right;
keyshift[107] = XK_End;
keyshift[108] = XK_Down;
keyshift[109] = XK_Next;
keyshift[110] = XK_Insert;
keyshift[111] = XK_Delete;
keyshift[112] = NoSymbol;
```

```
keynorm[113] = NoSymbol;                keyshift[113] = NoSymbol;
keynorm[114] = NoSymbol;                keyshift[114] = NoSymbol;
keynorm[115] = NoSymbol;                keyshift[115] = NoSymbol;
keynorm[116] = NoSymbol;                keyshift[116] = NoSymbol;
keynorm[117] = NoSymbol;                keyshift[117] = NoSymbol;
keynorm[118] = NoSymbol;                keyshift[118] = NoSymbol;
keynorm[119] = XK_Pause;                keyshift[119] = XK_Pause;
keynorm[120] = NoSymbol;                keyshift[120] = NoSymbol;
keynorm[121] = NoSymbol;                keyshift[121] = NoSymbol;
keynorm[122] = NoSymbol;                keyshift[122] = NoSymbol;
keynorm[123] = NoSymbol;                keyshift[123] = NoSymbol;
keynorm[124] = NoSymbol;                keyshift[124] = NoSymbol;
keynorm[125] = NoSymbol;                keyshift[125] = NoSymbol;
keynorm[126] = NoSymbol;                keyshift[126] = NoSymbol;
keynorm[127] = NoSymbol;                keyshift[127] = NoSymbol;
```

ANEXO 22 Tipos de chips de memoria no-volatiles

PROM: Acronimo de Programable Read Only Memory, es un integrado en el cual se le pueden escribir datos una sola vez. Una vez que el programa se escribio queda para siempre. A diferencia de la RAM, la informacion alojada en la PROM se mantiene cuando es interrumpida la electricidad. La diferencia entre una PROM y una ROM, es que la primera se vende en blanco, mientras que la ROM se vende ya programada. Para programar una PROM necesias un aparato especial llamado "programador PROM o quemador PROM". El proceso de programacion tambien es llamado muchas veces "quemado". Una variante de PROM es la EPROM, la cual se le puede borrar su informacion exponiendola a la luz ultravioleta, una vez que ha sido borrada, puede ser reprogramada. Una EEPROM es similiar es similar a la PROM, pero requiere solo electricidad para borrar su contenido.

EPROM: Acronimo de Erasable Programable Read Only Memory, la cual es un tipo especial de memoria que mantiene su contenido hasta que se la exponga a luz ultravioleta. Este tipo de luz limpia su contenido, haciendo posible reprogramar la memoria, para escribir este tipo de memoria puedes utilizar el mismo tipo de programador que para las PROM. La tecnologia de las EPROMS se distinguen porque su programacion se basa en la inyeccion de electrones con un elevado voltaje dentro de las gates de transistores de efecto de campo (FETs) cuando se desea un bit 0. Los electrones asi atrapados, logran que el transistor conduzca, leyendose como 0. Para borrar la EPROM, los electrones atrapados obtienen suficiente energia para liberarse de la compuerta (gate) por el constante bombardeo de radiacion ultravioleta detras de una ventana de Cuarzo. Para evitar el lento borrado natural a traves de los años, ya sea por la luz del sol u otras fuentes, esta ventana es tapada con una etiqueta.

EEPROM: Acronimo de Electrically Erasable Programmable Read Only Memory. Una EEPROM es un tipo especial de memoria PROM que puede ser borrada al exponerla a cargas electricas. Asi como las demas PROMs, esta memoria tambien mantiene la informacion cuando la electricidad se interrumpe. Al igual que las demas PROM, esta memoria tampoco es tan rapida como las memorias RAM. EEPROM es similar a las memorias Flash. La diferencia entre ambas es que mientras que la EEPROM necesita ser escrita o borrada byte a byte, la memoria Flash permite ser escrita o borrada en bloques. Esoto hace a la memoria Flash mas rapida.

FRAM: Acronimo de Ferroelectric Random Access Memory, un tipo de memoria no volatil desarrollada por Ramtron International Corporation. FRAM combina la velocidad de acceso de una DRAM o SRAM con la no volatibilidad de una PROM. A causa de su alta velocidad esta reemplazando a las memorias EEPROM en muchos dispositivos. El termino FRAM en si mismo es una marca registrada de Ramtron.

NVRAM: Acronimo de Non Volatile Random Access Memory, un tipo de memoria que mantiene la informacion cuando la electricidad se interrumpe. Un tipo de NVRAM es la SRAM, la cual mantiene la informacion gracias a un constante flujo de electricidad como el de una pila o bateria. Otros tipos de NVRAMs utilizan EEPROMS para mantener los datos despues de la interrupcion de la energia electrica. En estos casos, estas NVRAMs son el resultado de combinar SRAMs y EEPROMs.

Memorias de burbuja: Es un tipo de memoria no volatil compuesta por una fina capa de material que puede ser facilmente magnetizable en una sola direccion. Cuando un campo magnetico es aplicado a un area circular de esta sustancia, este campo se cierra en un pequeño circulo o burbuja. De todas formas, una EEPROM aparentemente es mas rapida y mas barata que este tipo de

tecnologia todavia en fase de desarrollo.

Memoria Flash: Es un tipo especial de EEPROM que puede ser borrada y reprogramada en bloques en vez de un byte a la vez. Muchas Pcs modernas tienen alojadas sus BIOS en este tipo de memorias, obteniendo la ventaja de poder ser actualizadas facilmente si es necesario. Asi tambien a veces las BIOS son tambien llamadas Flash-BIOS. Estas memorias han sido muy populares en los modems, porque habilitaba a los creadores de estos modems soportar nuevos protocolos mientras iban apareciendo.

Extraido y traducido del articulo "Network Boot and Exotic Root HOWTO"

ANEXO 23 Lista de NICs (placas de red) soportadas por EtherBoot:

```
familia     drivers/net/skel
skel-pci    0x0000,0x0000     Skeleton PCI Adaptor
skel-isa    -     Skeleton ISA driver

familia     drivers/net/3c595
3c590 0x10b7,0x5900     3Com590
3c595 0x10b7,0x5950     3Com595
3c595-1     0x10b7,0x5951     3Com595
3c595-2     0x10b7,0x5952     3Com595
3c900-tpo   0x10b7,0x9000     3Com900-TPO
3c900-t4    0x10b7,0x9001     3Com900-Combo
3c900b-tpo  0x10b7,0x9004     3Com900B-TPO
3c900b-combo      0x10b7,0x9005     3Com900B-Combo
3c900b-tpb2 0x10b7,0x9006     3Com900B-2/T
3c900b-fl   0x10b7,0x900a     3Com900B-FL
3c980-cyclone-1   0x10b7,0x9800     3Com980-Cyclone
3c9805-1    0x10b7,0x9805     3Com9805
3csoho100-tx-1    0x10b7,0x7646     3CSOHO100-TX
3c450-1     0x10b7,0x4500     3Com450 HomePNA Tornado

familia     drivers/net/3c90x
3c905-tpo   0x10b7,0x9000     3Com900-TPO
3c905-t4    0x10b7,0x9001     3Com900-Combo
3c905-tpo100      0x10b7,0x9050     3Com905-TX
3c905-combo 0x10b7,0x9051     3Com905-T4
3c905b-tpo  0x10b7,0x9004     3Com900B-TPO
3c905b-combo      0x10b7,0x9005     3Com900B-Combo
3c905b-tpb2 0x10b7,0x9006     3Com900B-2/T
3c905b-fl   0x10b7,0x900a     3Com900B-FL
3c905b-tpo100     0x10b7,0x9055     3Com905B-TX
3c905b-t4   0x10b7,0x9056     3Com905B-T4
3c905b-9058 0x10b7,0x9058     3Com905B-9058
3c905b-fx   0x10b7,0x905a     3Com905B-FL
3c905c-tpo  0x10b7,0x9200     3Com905C-TXM
3c980 0x10b7,0x9800     3Com980-Cyclone
3c9805      0x10b7,0x9805     3Com9805
3csoho100-tx      0x10b7,0x7646     3CSOHO100-TX
3c450 0x10b7,0x4500     3Com450 HomePNA Tornado

familia     drivers/net/eepro100
id1029      0x8086,0x1029     Intel EtherExpressPro100 ID1029
id1030      0x8086,0x1030     Intel EtherExpressPro100 ID1030
82801cam    0x8086,0x1031     Intel 82801CAM (ICH3) Chipset Ethernet Controller
eepro100-1032     0x8086,0x1032     Intel PRO/100 VE Network Connection
eepro100-1033     0x8086,0x1033     Intel PRO/100 VM Network Connection
eepro100-1034     0x8086,0x1034     Intel PRO/100 VM Network Connection
eepro100-1035     0x8086,0x1035     Intel 82801CAM (ICH3) Chipset Ethernet
Controller
eepro100-1036     0x8086,0x1036     Intel 82801CAM (ICH3) Chipset Ethernet
Controller
eepro100-1037     0x8086,0x1037     Intel 82801CAM (ICH3) Chipset Ethernet
Controller
id1038      0x8086,0x1038     Intel PRO/100 VM Network Connection
82562et     0x8086,0x1039     Intel PRO100 VE 82562ET
id103a      0x8086,0x103a     Intel Corporation 82559 InBusiness 10/100
82562etb    0x8086,0x103b     Intel PRO100 VE 82562ETB
eepro100-103c     0x8086,0x103c     Intel PRO/100 VM Network Connection
```

```
eepro100-103d     0x8086,0x103d     Intel PRO/100 VE Network Connection
eepro100-103e     0x8086,0x103e     Intel PRO/100 VM Network Connection
82551qm     0x8086,0x1059     Intel PRO/100 M Mobile Connection
82559er     0x8086,0x1209     Intel EtherExpressPro100 82559ER
82865 0x8086,0x1227     Intel 82865 EtherExpress PRO/100A
82556 0x8086,0x1228     Intel 82556 EtherExpress PRO/100 Smart
eepro100    0x8086,0x1229     Intel EtherExpressPro100
82562em     0x8086,0x2449     Intel EtherExpressPro100 82562EM
82562-1     0x8086,0x2459     Intel 82562 based Fast Ethernet Connection
82562-2     0x8086,0x245d     Intel 82562 based Fast Ethernet Connection
82562ez     0x8086,0x1050     Intel 82562EZ Network Connection
eepro100-5200     0x8086,0x5200     Intel EtherExpress PRO/100 Intelligent
Server
eepro100-5201     0x8086,0x5201     Intel EtherExpress PRO/100 Intelligent
Server

familia     drivers/net/e1000
e1000-82542 0x8086,0x1000     Intel EtherExpressPro1000
e1000-82543gc-fiber     0x8086,0x1001     Intel EtherExpressPro1000 82543GC
Fiber
e1000-82543gc-copper    0x8086,0x1004     Intel EtherExpressPro1000 82543GC
Copper
e1000-82544ei-copper    0x8086,0x1008     Intel EtherExpressPro1000 82544EI
Copper
e1000-82544ei-fiber     0x8086,0x1009     Intel EtherExpressPro1000 82544EI
Fiber
e1000-82544gc-copper    0x8086,0x100c     Intel EtherExpressPro1000 82544GC
Copper
e1000-82544gc-lom 0x8086,0x100d     Intel EtherExpressPro1000 82544GC LOM
e1000-82540em     0x8086,0x100e     Intel EtherExpressPro1000 82540EM
e1000-82545em-copper    0x8086,0x100f     Intel EtherExpressPro1000 82545EM
Copper
e1000-82546eb-copper    0x8086,0x1010     Intel EtherExpressPro1000 82546EB
Copper
e1000-82545em-fiber     0x8086,0x1011     Intel EtherExpressPro1000 82545EM
Fiber
e1000-82546eb-fiber     0x8086,0x1012     Intel EtherExpressPro1000 82546EB
Copper
e1000-82541ei     0x8086,0x1013     Intel EtherExpressPro1000 82541EI
e1000-82540em-lom 0x8086,0x1015     Intel EtherExpressPro1000 82540EM LOM
e1000-82540ep-lom 0x8086,0x1016     Intel EtherExpressPro1000 82540EP LOM
e1000-82540ep     0x8086,0x1017     Intel EtherExpressPro1000 82540EP
e1000-82541ep     0x8086,0x1018     Intel EtherExpressPro1000 82541EP
e1000-82547ei     0x8086,0x1019     Intel EtherExpressPro1000 82547EI
e1000-82546eb-quad-copper     0x8086,0x101d     Intel EtherExpressPro1000 82546EB
Quad Copper
e1000-82540ep-lp  0x8086,0x101e     Intel EtherExpressPro1000 82540EP LP

familia     drivers/net/tg3
tg3-5700    0x14e4,0x1644     Broadcom Tigon 3 5700
tg3-5701    0x14e4,0x1645     Broadcom Tigon 3 5701
tg3-5702    0x14e4,0x1646     Broadcom Tigon 3 5702
tg3-5703    0x14e4,0x1647     Broadcom Tigon 3 5703
tg3-5704    0x14e4,0x1648     Broadcom Tigon 3 5704
tg3-5702FE  0x14e4,0x164d     Broadcom Tigon 3 5702FE
tg3-5705    0x14e4,0x1653     Broadcom Tigon 3 5705
tg3-5705_2  0x14e4,0x1654     Broadcom Tigon 3 5705_2
tg3-5705M   0x14e4,0x165d     Broadcom Tigon 3 5705M
tg3-5705M_2 0x14e4,0x165e     Broadcom Tigon 3 5705M_2
tg3-5782    0x14e4,0x1696     Broadcom Tigon 3 5782
tg3-5788    0x14e4,0x169c     Broadcom Tigon 3 5788
```

```
tg3-5702X   0x14e4,0x16a6     Broadcom Tigon 3 5702X
tg3-5703X   0x14e4,0x16a7     Broadcom Tigon 3 5703X
tg3-5704S   0x14e4,0x16a8     Broadcom Tigon 3 5704S
tg3-5702A3  0x14e4,0x16c6     Broadcom Tigon 3 5702A3
tg3-5703A3  0x14e4,0x16c7     Broadcom Tigon 3 5703A3
tg3-5901    0x14e4,0x170d     Broadcom Tigon 3 5901
tg3-5901_2  0x14e4,0x170e     Broadcom Tigon 3 5901_2
tg3-9DXX    0x1148,0x4400     Sysconnet 9DXX
tg3-9MXX    0x1148,0x4500     Sysconnet 9MXX
tg3-ac1000  0x173b,0x03e8     Altima AC1000
tg3-ac1001  0x173b,0x03e9     Altima AC1001
tg3-ac9100  0x173b,0x03ea     Altima AC9100
tg3-ac1003  0x173b,0x03eb     Altima AC1003

familia     drivers/net/pcnet32
lancepci    0x1022,0x2000     AMD Lance/PCI
pcnetfastiii      0x1022,0x2625     AMD Lance/PCI PCNet/32
amdhomepna  0x1022,0x2001     AMD Lance/HomePNA

familia     drivers/net/tulip
dc21040     0x1011,0x0002     Digital Tulip
ds21140     0x1011,0x0009     Digital Tulip Fast
dc21041     0x1011,0x0014     Digital Tulip+
ds21142     0x1011,0x0019     Digital Tulip 21142
mx98713     0x10d9,0x0512     Macronix MX987x3
mx98715     0x10d9,0x0531     Macronix MX987x5
mxic-98715  0x1113,0x1217     Macronix MX987x5
lc82c115    0x11ad,0xc115     LinkSys LNE100TX
82c168      0x11ad,0x0002     Netgear FA310TX
dm9100      0x1282,0x9100     Davicom 9100
dm9102      0x1282,0x9102     Davicom 9102
dm9009      0x1282,0x9009     Davicom 9009
centaur-p   0x1317,0x0985     ADMtek Centaur-P
an981 0x1317,0x0981     ADMtek AN981 Comet
an983 0x1113,0x1216     ADMTek AN983 Comet
an983b      0x1317,0x9511     ADMTek Comet 983b
centaur-c   0x1317,0x1985     ADMTek Centaur-C
intel21145  0x8086,0x0039     Intel Tulip
ax88140     0x125b,0x1400     ASIX AX88140
rl100tx     0x11f6,0x9881     Compex RL100-TX
xircomtulip 0x115d,0x0003     Xircom Tulip
tulip-0981  0x104a,0x0981     Tulip 0x104a 0x0981
tulip-2774  0x104a,0x2774     Tulip 0x104a 0x2774
tulip-9511  0x1113,0x9511     Tulip 0x1113 0x9511
tulip-1561  0x1186,0x1561     Tulip 0x1186 0x1561
tulip-a120  0x1259,0xa120     Tulip 0x1259 0xa120
tulip-ab02  0x13d1,0xab02     Tulip 0x13d1 0xab02
tulip-ab03  0x13d1,0xab03     Tulip 0x13d1 0xab03
tulip-ab08  0x13d1,0xab08     Tulip 0x13d1 0xab08
lanfinity   0x14f1,0x1803     Conexant LANfinity
tulip-8410  0x1626,0x8410     Tulip 0x1626 0x8410
tulip-ab09  0x1737,0xab09     Tulip 0x1737 0xab09

familia     drivers/net/davicom
davicom9100 0x1282,0x9100     Davicom 9100
davicom9102 0x1282,0x9102     Davicom 9102
davicom9009 0x1282,0x9009     Davicom 9009
davicom9132 0x1282,0x9132     Davicom 9132

familia     drivers/net/rtl8139
rtl8129     0x10ec,0x8129     Realtek 8129
```

```
rtl8139     0x10ec,0x8139     Realtek 8139
rtl8139b    0x10ec,0x8138     Realtek 8139B
dfe538      0x1186,0x1300     DFE530TX+/DFE538TX
smc1211-1   0x1113,0x1211     SMC EZ10/100
smc1211     0x1112,0x1211     SMC EZ10/100
delta8139   0x1500,0x1360     Delta Electronics 8139
addtron8139 0x4033,0x1360     Addtron Technology 8139
dfe690txd   0x1186,0x1340     D-Link DFE690TXD
fe2000vx    0x13d1,0xab06     AboCom FE2000VX
allied8139  0x1259,0xa117     Allied Telesyn 8139
fnw3603tx   0x14ea,0xab06     Planex FNW-3603-TX
fnw3800tx   0x14ea,0xab07     Planex FNW-3800-TX
clone-rtl8139     0xffff,0x8139     Cloned 8139

familia     drivers/net/via-rhine
dlink-530tx 0x1106,0x3065     VIA 6102
via-rhine-6105    0x1106,0x3106     VIA 6105
dlink-530tx-old   0x1106,0x3043     VIA 3043
via6105m    0x1106,0x3053     VIA 6105M
via-rhine-old     0x1106,0x6100     VIA 86C100A

familia     drivers/net/w89c840
winbond840  0x1050,0x0840     Winbond W89C840F
compexrl100atx    0x11f6,0x2011     Compex RL100ATX

familia     drivers/net/sis900
sis900      0x1039,0x0900     SIS900
sis7016     0x1039,0x7016     SIS7016

familia     drivers/net/natsemi
dp83815     0x100b,0x0020     DP83815

familia     drivers/net/fa311
fa311too    0x100b,0x0020     DP83815

familia     drivers/net/prism2_plx
ma301 0x1385,0x4100     Netgear MA301
3c-airconnect     0x10b7,0x7770     3Com AirConnect
ss1023      0x111a,0x1023     Siemens SpeedStream SS1023
correga     0x15e8,0x0130     Correga
smc2602w    0x1638,0x1100     SMC EZConnect SMC2602W
gl24110p    0x16ab,0x1100     Global Sun Tech GL24110P
16ab-1101   0x16ab,0x1101     Unknown
wdt11 0x16ab,0x1102     Linksys WDT11
usr2415     0x16ec,0x3685     USR 2415
f5d6000     0xec80,0xec00     Belkin F5D6000
emobility   0x126c,0x8030     Nortel emobility

familia     drivers/net/prism2_pci
prism2_pci  0x1260,0x3873     Harris Semiconductor Prism2.5 clone
hwp01170    0x1260,0x3873     ActionTec HWP01170
dwl520      0x1260,0x3873     DLink DWL-520

familia     drivers/net/ns8390
rtl8029     0x10ec,0x8029     Realtek 8029
dlink-528   0x1186,0x0300     D-Link DE-528
winbond940  0x1050,0x0940     Winbond NE2000-PCI
winbond940f 0x1050,0x5a5a     Winbond W89c940F
compexrl2000      0x11f6,0x1401     Compex ReadyLink 2000
ktiet32p2   0x8e2e,0x3000     KTI ET32P2
nv5000sc    0x4a14,0x5000     NetVin NV5000SC
```

```
holtek80232 0x12c3,0x0058     Holtek HT80232
holtek80229 0x12c3,0x5598     Holtek HT80229
surecom-ne34      0x10bd,0x0e34     Surecom NE34
via86c926   0x1106,0x0926     Via 86c926
wd    -     WD8003/8013, SMC8216/8416, SMC 83c790 (EtherEZ)
ne    -     NE1000/2000 and clones
3c503 -     3Com503, Etherlink II[/16]

familia     drivers/net/epic100
epic100     0x10b8,0x0005     SMC EtherPowerII
smc-83c175  0x10b8,0x0006     SMC EPIC/C 83c175

familia     drivers/net/3c509
3c509 -     3c509, ISA/EISA
3c529 -     3c529 == MCA 3c509

familia     drivers/net/3c515
3c515 -     3c515, Fast EtherLink ISA

familia     drivers/net/eepro
eepro -     Intel Etherexpress Pro/10

familia     drivers/net/cs89x0
cs89x0      -     Crystal Semiconductor CS89x0

familia     drivers/net/depca
depca -     Digital DE100 and DE200

familia     drivers/net/sk_g16
sk_g16      -     Schneider and Koch G16

familia     drivers/net/smc9000
smc9000     -     SMC9000

familia     drivers/net/sundance
sundance    0x13f0,0x0201     ST201 Sundance 'Alta' based Adaptor
dfe530txs   0x1186,0x1002     D-Link DFE530TXS (Sundance ST201 Alta)

familia     drivers/net/tlan
netel10     0x0e11,0xae34     Compaq Netelligent 10 T PCI UTP
netel100    0x0e11,0xae32     Compaq Netelligent 10/100 TX PCI UTP
netflex3i   0x0e11,0xae35     Compaq Integrated NetFlex-3/P
thunder     0x0e11,0xf130     Compaq NetFlex-3/P
netflex3b   0x0e11,0xf150     Compaq NetFlex-3/P
netel100pi  0x0e11,0xae43     Compaq Netelligent Integrated 10/100 TX UTP
netel100d   0x0e11,0xae40     Compaq Netelligent Dual 10/100 TX PCI UTP
netel100i   0x0e11,0xb011     Compaq Netelligent 10/100 TX Embedded UTP
oc2183      0x108d,0x0013     Olicom OC-2183/2185
oc2325      0x108d,0x0012     Olicom OC-2325
oc2326      0x108d,0x0014     Olicom OC-2326
netelligent_10_100_ws_5100    0x0e11,0xb030     Compaq Netelligent 10/100 TX UTP
netelligent_10_t2 0x0e11,0xb012     Compaq Netelligent 10 T/2 PCI UTP/Coax

familia     drivers/disk/ide_disk
ide_disk    0x0000,0x0000     Generic IDE disk support

familia     drivers/disk/pc_floppy
pc_floppy   -     Generic PC Floppy support
```

ANEXO 24 Proceso de arranque del Kernel Linux

La cadena de eventos en el arranque son los siguientes:

CPU > VGA > AUTOCOMPROBACION DE ARRANQUE > SCSI/IDE > BOOT MANAGER > CARGADOR DE ARRANQUE LILO/GRUB > KERNEL > INIT > BASH

Un tour guiado por el proceso de arranque del Kernel:

1. La BIOS de la motherboard dispara la inicializacion de la placa de video
2. Ahora la BIOS se inicializa a si misma
3. Los controladores SCSI/IDE se inicializan
4. Resumen de Hardware: la BIOS muestra en pantalla un resumen del Hardware instalado en la maquina y corre su codigo antivirus que se basa en verificar si han cambiado los sectores de arranque del disco.
5. BootManager Menu : El Master Boot Record (MBR) del primer disco duro es leido, por tradicion a los viejos sistemas DOS en la direccion 0x00007c00, y el procesador arranca ejecutando instrucciones aquí. Este codigo de arranque MBR carga el primer sector de codigo de la particion DOS activa.
6. (Ejemplo con LILO): Lilo arranca: Si se elige la opcion LINUX y ademas Linux ah sido instalado con lilo, lilo es cargado en la direccion 0x00007c00. Lilo imprime LILO progresando letra a letra. La primer "L" es impresa despues de que lilo se mueve el mismo a una mejor ubicación como 0x0009A000. La "I" es impresa justo antes de que comience a ejecutarse el codigo del segundo cargador de arranque. El segundo cargador de arranque de liloimprime la proxima "L", Carga ahora las tablas descriptoras que apuntan a partes del Kernel, y luego imprime la "O" final. Los descriptores estan ubicados en la direccion 0x0009d200. Se puede ver el mensage de booteo y una linea de prompt, la cual si ha sido especificada , se imprime en pantalla. Presionando "Tab" en el prompt, permite al usuario especificar un sistema y proveer especificaciones en formato linea de comandos al Kernel, sus drivers por ejemplo, y el programa "init". Tambien variables de entorno pueden ser definidas en este punto.

   Las siguientes lineas provienen de /boot/message:

```
>
>
>
 Press  to list available boot image labels.
The following line is the prompt from /sbin/lilo:
boot:
Note: If Lilo is not used, then the boot code built into the head
      of the Linux kernel, linux/arch/i386/boot/bootsect.S
          prints "Loading" and continues.
Lilo displays the following as it loads the kernel code. It gets the
text "Linux-2.2.12" from the "label=..." specification in lilo.conf.
Loading linux-2.2.12.........
```

7. El código del Kernel en /linux/arch/i386/boot/setup.S arregla la transicion de arranque en modo real (DOS mode) al modo protegido (full 32-bit mode). Bloques de codigo llamados Trampoline.S y Trampoline32.S ayudan con la transicion. Pequeñas imagenes del Kernel (zImage) son descomprimidas y cargadas en la direccion de memoria 0x00010000. Imagenes grandes de Kernel (bzImage) son cargadas en 0x00100000. Este codigo levanta los registros, descomprime el kernel comprimido (el cual esta en linux/arch/i386/head.S al principio), imprimiendo las siguientes dos lineas de linux/arch/i386/boot/compressed/misc.c

`Uncompressing Linux... Ok. Booting the kernel`. El codigo especifico para i386 setup.S ahora ha completado su trabajo y salta a 0x00010000 (o 0x00100000) para arrancar el codigo del Kernel generico.

a) Inicializacion del procesador, consola y memoria: Esto corre linux/arch/i386/head.S el cual salta a start_kernel(void) en linux/init/main.c donde las interrupciones son redefinida. linux/kernel/module.c luego carga los drivers para la consola y el BUS PCI. A partir de este punto, los mensajes del Kernel son guardados en memoria y disponibles usando /bin/dmesg. Parte de estos mensajes son transferidos a /var/log/message para una grabacion permanente.

b)Inicializacion del BUS PCI: mpci_init() en linux/init/main.c genera lineas desde linux/arch/i386/kernel/bios32.c para ser impresas.

c) Inicializacion de la RED: socket_init() en linux/init/main.c genera la siguiente inicializacion de red:
linux/net/socket.c imprime:

```
Linux NET4.0 for Linux 2.2
Based upon Swansea University Computer Society NET3.039
linux/net/unix/af_unix.c prints:
NET4: Unix domain sockets 1.0 for Linux NET4.0.
linux/net/ipv4/af_inet.c prints:
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP
linux/net/ipv4/ip_gre.c prints:
GRE over IPv4 tunneling driver
linux/net/core/dev.c prints:
early initialization of device gre0 is deferred
linux/net/core/rtnetlink.c prints:
Initializing RT netlink socket
```

d) El hilo Idle del Kernel (Proceso 0) arranca: En este punto, un hilo del Kernel se inicia corriendo init() la cual es una de las rutinas definidas en linux/init/main.c. Esta init() no debe ser confundida con el programa /sbin/init el cual correra despues de que el Kernel este levantado y corriendo. Ahora, mkswapd_setup() en linux/init/main.c genera la siguiente linea desde linux/mm/vmscan.c para ser impresa: `Starting kswapd v 1.5`

e) Initialization de los dispositivos: la rutina del Kernel alojada en linux/arch/i386/kernel/setup.c ahora inicializa los dispositivos y sistemas de archivos (built into the kernel??). Produce lineas en pantalla y fuerza a correr /sbin/init

e1) Rutina de inicializacion del Port paralelo generico: linux/drivers/misc/parport_pc.c

e2) Inicializacion de los dispositivos de caracteres desde linux/drivers/char/serial.c

e3) Inicializacion de los dispositivos de bloques: linux/drivers/block/rd.c imprime: `RAM disk driver initialized: 16 RAM disks of 8192K size` linux/drivers/block/loop.c imprime: `loop: registered device at major 7` linux/drivers/block/floppy.c imprime: `Floppy drive(s): fd0 is 1.44M, fd1 is 1.44M FDC 0 is a post-1991 82077`

e4) Inicializacion IDE/SCSI: Proveniente de aic7xxx.c, scsi.c, sg.c, sd.c o sr.c en el subdirectorio linux/drivers/scsi.

f) Initializacion de soporte PPP (Point-to-Point Protocol) para el Kernel. La inicializacion se hace en linux/drivers/net/ppp.c.

g) Examen de arreglo de disco fijo: Viene de linux/drivers/block/genhd.c

8. Programa Init (Proceso 1) Arranque: El programa /sbin/init arranca gracias al proceso "idle" (Proceso 0) en linux/init/main.c y se convierte en el proceso 1. /sbin/init entonces completa la initializacion corriendo scripts y forzando procesos adicionales que han sido especificados

en /etc/inittab. Comienza imprimiendo: `INIT: version 2.76 booting` y luego lee /etc/inittab.

9. Bash arranca: El shell bash, /bin/bash es ahora inicializado. La inicializacion de Bash comienza ejecutando el script el /etc/profile el cual setea las variables de sistema

(Extraido de The Linux Kernel HOWTO - Al Dev (Alavoor Vasudevan)

ANEXO 25 Guía rápida para comprimir y descomprimir archivos

Copyright 2005-2008 Sergio González Durán
Se concede permiso para copiar, distribuir y/o modificar este documento siempre y cuando se cite al autor y la fuente de linuxtotal.com.mx y según los términos de la GNU Free Documentation License, Versión 1.2 o cualquiera posterior publicada por la Free Software Foundation.
autor: sergio.gonzalez.duran@gmail.com

En Linux hay diversas herramientas para empaquetar y comprimir archivos, tomando en cuenta que empaquetar es juntar dos o más archivos en un solo archivo (paquete) y comprimir es tomar este archivo-paquete y comprimirlo a continuación te muestro un resumen de las más comunes, de acuerdo a la extensión que comunmente se acostumbra ponerles.

.tar (tar)	
Empaquetar	tar cvf archivo.tar /archivo/mayo/*
Desempaquetar	tar xvf archivo.tar
Ver el contenido (sin extraer)	tar tvf archivo.tar
.tar.gz - .tar.z - .tgz (tar con gzip)	
Empaquetar y comprimir	tar czvf archivo.tar.gz /archivo/mayo/*
Desempaquetar y descomprimir	tar xzvf archivo.tar.gz
Ver el contenido (sin extraer)	tar tzvf archivo.tar.gz
.gz (gzip)	
Comprimir	gzip -q archivo (El archivo lo comprime y lo renombra como "archivo.gz")
Descomprimir	gzip -d archivo.gz (El archivo lo descomprime y lo deja como "archivo"
Nota: gzip solo comprime archivos, no directorios	
.bz2 (bzip2)	
Comprimir	bzip2 archivo bunzip2 archivo (El archivo lo comprime y lo renombra como "archivo.bz2")
Descomprimir	bzip2 -d archivo.bz2 bunzip2 archivo.bz2 (El archivo lo descomprime y lo deja como "archivo")
Nota: bzip2 solo comprime archivos, no directorios	
.tar.bz2 (tar con bzip2)	
Comprimir	tar -c archivos | bzip2 > archivo.tar.bz2
Descomprimir	bzip2 -dc archivo.tar.bz2 | tar -xv tar jvxf archivo.tar.bz2 (versiones recientes de tar)
Ver contenido	bzip2 -dc archivo.tar.bz2 | tar -tv

.zip (zip)	
Comprimir	zip archivo.zip /mayo/archivos
Descomprimir	unzip archivo.zip
Ver contenido	unzip -v archivo.zip
.lha (lha)	
Comprimir	lha archivo.lha /mayo/archivos
Descomprimir	lha -x archivo.lha
Ver contenido	lha -v archivo.lha lha -l archivo.lha
.zoo (zoo)	
Comprimir	zoo -a archivo.zoo /mayo/archivos
Descomprimir	zoo -x archivo.zoo
Ver contenido	zoo -v archivo.zoo zoo -L archivo.zoo
.rar (rar)	
Comprimir	rar -a archivo.rar /mayo/archivos
Descomprimir	rar -x archivo.rar
Ver contenido	rar -v archivo.rar rar -l archivo.rar

ANEXO 26 Mi .config del Kernel 2.4.22

```
#
# Automatically generated by make menuconfig: don't edit
#
CONFIG_X86=y
# CONFIG_SBUS is not set
CONFIG_UID16=y

#
# Code maturity level options
#
CONFIG_EXPERIMENTAL=y

#
# Loadable module support
#
CONFIG_MODULES=y
# CONFIG_MODVERSIONS is not set
CONFIG_KMOD=y

#
# Processor type and features
#
# CONFIG_M386 is not set
CONFIG_M486=y
# CONFIG_M586 is not set
# CONFIG_M586TSC is not set
# CONFIG_M586MMX is not set
# CONFIG_M686 is not set
# CONFIG_MPENTIUMIII is not set
# CONFIG_MPENTIUM4 is not set
# CONFIG_MK6 is not set
# CONFIG_MK7 is not set
# CONFIG_MK8 is not set
# CONFIG_MELAN is not set
# CONFIG_MCRUSOE is not set
# CONFIG_MWINCHIPC6 is not set
# CONFIG_MWINCHIP2 is not set
# CONFIG_MWINCHIP3D is not set
# CONFIG_MCYRIXIII is not set
# CONFIG_MVIAC3_2 is not set
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_CMPXCHG=y
CONFIG_X86_XADD=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
# CONFIG_RWSEM_GENERIC_SPINLOCK is not set
CONFIG_RWSEM_XCHGADD_ALGORITHM=y
CONFIG_X86_L1_CACHE_SHIFT=4
CONFIG_X86_USE_STRING_486=y
CONFIG_X86_ALIGNMENT_16=y
CONFIG_X86_PPRO_FENCE=y
# CONFIG_X86_F00F_WORKS_OK is not set
# CONFIG_X86_MCE is not set
# CONFIG_TOSHIBA is not set
# CONFIG_I8K is not set
# CONFIG_MICROCODE is not set
# CONFIG_X86_MSR is not set
# CONFIG_X86_CPUID is not set
CONFIG_NOHIGHMEM=y
# CONFIG_HIGHMEM4G is not set
# CONFIG_HIGHMEM64G is not set
# CONFIG_HIGHMEM is not set
CONFIG_MATH_EMULATION=y
# CONFIG_MTRR is not set
# CONFIG_SMP is not set
```

```
# CONFIG_X86_UP_APIC is not set
# CONFIG_X86_UP_IOAPIC is not set
# CONFIG_X86_TSC_DISABLE is not set

#
# General setup
#
CONFIG_NET=y
CONFIG_PCI=y
# CONFIG_PCI_GOBIOS is not set
# CONFIG_PCI_GODIRECT is not set
CONFIG_PCI_GOANY=y
CONFIG_PCI_BIOS=y
CONFIG_PCI_DIRECT=y
CONFIG_ISA=y
# CONFIG_PCI_NAMES is not set
# CONFIG_EISA is not set
# CONFIG_MCA is not set
# CONFIG_HOTPLUG is not set
# CONFIG_PCMCIA is not set
# CONFIG_HOTPLUG_PCI is not set
# CONFIG_SYSVIPC is not set
# CONFIG_BSD_PROCESS_ACCT is not set
# CONFIG_SYSCTL is not set
# CONFIG_BINFMT_AOUT is not set
CONFIG_BINFMT_ELF=y
# CONFIG_BINFMT_MISC is not set
# CONFIG_PM is not set
# CONFIG_APM is not set

#
# ACPI Support
#
# CONFIG_ACPI is not set

#
# Memory Technology Devices (MTD)
#
# CONFIG_MTD is not set

#
# Parallel port support
#
# CONFIG_PARPORT is not set

#
# Plug and Play configuration
#
# CONFIG_PNP is not set
# CONFIG_ISAPNP is not set

#
# Block devices
#
# CONFIG_BLK_DEV_FD is not set
# CONFIG_BLK_DEV_XD is not set
# CONFIG_PARIDE is not set
# CONFIG_BLK_CPQ_DA is not set
# CONFIG_BLK_CPQ_CISS_DA is not set
# CONFIG_CISS_SCSI_TAPE is not set
# CONFIG_BLK_DEV_DAC960 is not set
# CONFIG_BLK_DEV_UMEM is not set
# CONFIG_BLK_DEV_LOOP is not set
# CONFIG_BLK_DEV_NBD is not set
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_INITRD=y
# CONFIG_BLK_STATS is not set

#
```

```
# Multi-device support (RAID and LVM)
#
# CONFIG_MD is not set
# CONFIG_BLK_DEV_MD is not set
# CONFIG_MD_LINEAR is not set
# CONFIG_MD_RAID0 is not set
# CONFIG_MD_RAID1 is not set
# CONFIG_MD_RAID5 is not set
# CONFIG_MD_MULTIPATH is not set
# CONFIG_BLK_DEV_LVM is not set

#
# Networking options
#
# CONFIG_PACKET is not set
# CONFIG_NETLINK_DEV is not set
# CONFIG_NETFILTER is not set
# CONFIG_FILTER is not set
CONFIG_UNIX=y
CONFIG_INET=y
# CONFIG_IP_MULTICAST is not set
# CONFIG_IP_ADVANCED_ROUTER is not set
CONFIG_IP_PNP=y
CONFIG_IP_PNP_DHCP=y
CONFIG_IP_PNP_BOOTP=y
# CONFIG_IP_PNP_RARP is not set
# CONFIG_NET_IPIP is not set
# CONFIG_NET_IPGRE is not set
# CONFIG_ARPD is not set
# CONFIG_INET_ECN is not set
# CONFIG_SYN_COOKIES is not set
# CONFIG_IPV6 is not set
# CONFIG_KHTTPD is not set
# CONFIG_ATM is not set
# CONFIG_VLAN_8021Q is not set
# CONFIG_IPX is not set
# CONFIG_ATALK is not set

#
# Appletalk devices
#
# CONFIG_DEV_APPLETALK is not set
# CONFIG_DECNET is not set
# CONFIG_BRIDGE is not set
# CONFIG_X25 is not set
# CONFIG_LAPB is not set
# CONFIG_LLC is not set
# CONFIG_NET_DIVERT is not set
# CONFIG_ECONET is not set
# CONFIG_WAN_ROUTER is not set
# CONFIG_NET_FASTROUTE is not set
# CONFIG_NET_HW_FLOWCONTROL is not set

#
# QoS and/or fair queueing
#
# CONFIG_NET_SCHED is not set

#
# Network testing
#
# CONFIG_NET_PKTGEN is not set

#
# Telephony Support
#
# CONFIG_PHONE is not set
# CONFIG_PHONE_IXJ is not set
# CONFIG_PHONE_IXJ_PCMCIA is not set
```

```
#
# ATA/IDE/MFM/RLL support
#
CONFIG_IDE=y

#
# IDE, ATA and ATAPI Block devices
#
CONFIG_BLK_DEV_IDE=y
# CONFIG_BLK_DEV_HD_IDE is not set
# CONFIG_BLK_DEV_HD is not set
# CONFIG_BLK_DEV_IDEDISK is not set
# CONFIG_IDEDISK_MULTI_MODE is not set
# CONFIG_IDEDISK_STROKE is not set
# CONFIG_BLK_DEV_IDECS is not set
# CONFIG_BLK_DEV_IDECD is not set
# CONFIG_BLK_DEV_IDETAPE is not set
# CONFIG_BLK_DEV_IDEFLOPPY is not set
# CONFIG_BLK_DEV_IDESCSI is not set
# CONFIG_IDE_TASK_IOCTL is not set
# CONFIG_BLK_DEV_CMD640 is not set
# CONFIG_BLK_DEV_CMD640_ENHANCED is not set
# CONFIG_BLK_DEV_ISAPNP is not set
# CONFIG_BLK_DEV_IDEPCI is not set
# CONFIG_IDE_CHIPSETS is not set
CONFIG_IDEDMA_AUTO=y
# CONFIG_IDEDMA_IVB is not set
# CONFIG_DMA_NONPCI is not set
CONFIG_BLK_DEV_PDC202XX=y
CONFIG_BLK_DEV_IDE_MODES=y
# CONFIG_BLK_DEV_ATARAID is not set
# CONFIG_BLK_DEV_ATARAID_PDC is not set
# CONFIG_BLK_DEV_ATARAID_HPT is not set
# CONFIG_BLK_DEV_ATARAID_SII is not set

#
# SCSI support
#
# CONFIG_SCSI is not set

#
# Fusion MPT device support
#
# CONFIG_FUSION is not set
# CONFIG_FUSION_BOOT is not set
# CONFIG_FUSION_ISENSE is not set
# CONFIG_FUSION_CTL is not set
# CONFIG_FUSION_LAN is not set

#
# IEEE 1394 (FireWire) support (EXPERIMENTAL)
#
# CONFIG_IEEE1394 is not set

#
# I2O device support
#
# CONFIG_I2O is not set
# CONFIG_I2O_PCI is not set
# CONFIG_I2O_BLOCK is not set
# CONFIG_I2O_LAN is not set
# CONFIG_I2O_SCSI is not set
# CONFIG_I2O_PROC is not set

#
# Network device support
#
CONFIG_NETDEVICES=y

#
```

```
# ARCnet devices
#
# CONFIG_ARCNET is not set
# CONFIG_DUMMY is not set
# CONFIG_BONDING is not set
# CONFIG_EQUALIZER is not set
# CONFIG_TUN is not set
# CONFIG_ETHERTAP is not set

#
# Ethernet (10 or 100Mbit)
#
CONFIG_NET_ETHERNET=y
# CONFIG_SUNLANCE is not set
# CONFIG_HAPPYMEAL is not set
# CONFIG_SUNBMAC is not set
# CONFIG_SUNQE is not set
# CONFIG_SUNGEM is not set
# CONFIG_NET_VENDOR_3COM is not set
# CONFIG_LANCE is not set
# CONFIG_NET_VENDOR_SMC is not set
# CONFIG_NET_VENDOR_RACAL is not set
# CONFIG_AT1700 is not set
# CONFIG_DEPCA is not set
# CONFIG_HP100 is not set
CONFIG_NET_ISA=y
# CONFIG_E2100 is not set
# CONFIG_EWRK3 is not set
# CONFIG_EEXPRESS is not set
# CONFIG_EEXPRESS_PRO is not set
# CONFIG_HPLAN_PLUS is not set
# CONFIG_HPLAN is not set
# CONFIG_LP486E is not set
# CONFIG_ETH16I is not set
CONFIG_NE2000=y
# CONFIG_NET_PCI is not set
# CONFIG_NET_POCKET is not set

#
# Ethernet (1000 Mbit)
#
# CONFIG_ACENIC is not set
# CONFIG_DL2K is not set
# CONFIG_E1000 is not set
# CONFIG_MYRI_SBUS is not set
# CONFIG_NS83820 is not set
# CONFIG_HAMACHI is not set
# CONFIG_YELLOWFIN is not set
# CONFIG_R8169 is not set
# CONFIG_SK98LIN is not set
# CONFIG_TIGON3 is not set
# CONFIG_FDDI is not set
# CONFIG_HIPPI is not set
# CONFIG_PLIP is not set
# CONFIG_PPP is not set
# CONFIG_SLIP is not set

#
# Wireless LAN (non-hamradio)
#
# CONFIG_NET_RADIO is not set

#
# Token Ring devices
#
# CONFIG_TR is not set
# CONFIG_NET_FC is not set
# CONFIG_RCPCI is not set
# CONFIG_SHAPER is not set
```

```
#
# Wan interfaces
#
# CONFIG_WAN is not set

#
# Amateur Radio support
#
# CONFIG_HAMRADIO is not set

#
# IrDA (infrared) support
#
# CONFIG_IRDA is not set

#
# ISDN subsystem
#
# CONFIG_ISDN is not set

#
# Old CD-ROM drivers (not SCSI, not IDE)
#
# CONFIG_CD_NO_IDESCSI is not set

#
# Input core support
#
# CONFIG_INPUT is not set
# CONFIG_INPUT_KEYBDEV is not set
# CONFIG_INPUT_MOUSEDEV is not set
# CONFIG_INPUT_JOYDEV is not set
# CONFIG_INPUT_EVDEV is not set

#
# Character devices
#
CONFIG_VT=y
CONFIG_VT_CONSOLE=y
# CONFIG_SERIAL is not set
# CONFIG_SERIAL_EXTENDED is not set
# CONFIG_SERIAL_NONSTANDARD is not set
# CONFIG_UNIX98_PTYS is not set

#
# I2C support
#
# CONFIG_I2C is not set

#
# Mice
#
# CONFIG_BUSMOUSE is not set
CONFIG_MOUSE=y
CONFIG_PSMOUSE=y
# CONFIG_82C710_MOUSE is not set
# CONFIG_PC110_PAD is not set
# CONFIG_MK712_MOUSE is not set

#
# Joysticks
#
# CONFIG_INPUT_GAMEPORT is not set
# CONFIG_QIC02_TAPE is not set
# CONFIG_IPMI_HANDLER is not set
# CONFIG_IPMI_PANIC_EVENT is not set
# CONFIG_IPMI_DEVICE_INTERFACE is not set
# CONFIG_IPMI_KCS is not set
# CONFIG_IPMI_WATCHDOG is not set
```

```
#
# Watchdog Cards
#
# CONFIG_WATCHDOG is not set
# CONFIG_SCx200_GPIO is not set
# CONFIG_AMD_RNG is not set
# CONFIG_INTEL_RNG is not set
# CONFIG_AMD_PM768 is not set
# CONFIG_NVRAM is not set
# CONFIG_RTC is not set
# CONFIG_DTLK is not set
# CONFIG_R3964 is not set
# CONFIG_APPLICOM is not set
# CONFIG_SONYPI is not set

#
# Ftape, the floppy tape device driver
#
# CONFIG_FTAPE is not set
# CONFIG_AGP is not set
# CONFIG_DRM is not set
# CONFIG_MWAVE is not set

#
# Multimedia devices
#
# CONFIG_VIDEO_DEV is not set

#
# File systems
#
# CONFIG_QUOTA is not set
# CONFIG_QFMT_V2 is not set
CONFIG_AUTOFS_FS=y
# CONFIG_AUTOFS4_FS is not set
# CONFIG_REISERFS_FS is not set
# CONFIG_REISERFS_CHECK is not set
# CONFIG_REISERFS_PROC_INFO is not set
# CONFIG_ADFS_FS is not set
# CONFIG_ADFS_FS_RW is not set
# CONFIG_AFFS_FS is not set
# CONFIG_HFS_FS is not set
# CONFIG_HFSPLUS_FS is not set
# CONFIG_BEFS_FS is not set
# CONFIG_BEFS_DEBUG is not set
# CONFIG_BFS_FS is not set
# CONFIG_EXT3_FS is not set
# CONFIG_JBD is not set
# CONFIG_JBD_DEBUG is not set
# CONFIG_FAT_FS is not set
# CONFIG_MSDOS_FS is not set
# CONFIG_UMSDOS_FS is not set
# CONFIG_VFAT_FS is not set
# CONFIG_EFS_FS is not set
# CONFIG_JFFS_FS is not set
# CONFIG_JFFS2_FS is not set
# CONFIG_CRAMFS is not set
# CONFIG_TMPFS is not set
CONFIG_RAMFS=y
# CONFIG_ISO9660_FS is not set
# CONFIG_JOLIET is not set
# CONFIG_ZISOFS is not set
# CONFIG_JFS_FS is not set
# CONFIG_JFS_DEBUG is not set
# CONFIG_JFS_STATISTICS is not set
# CONFIG_MINIX_FS is not set
# CONFIG_VXFS_FS is not set
# CONFIG_NTFS_FS is not set
# CONFIG_NTFS_RW is not set
# CONFIG_HPFS_FS is not set
```

```
# CONFIG_PROC_FS is not set
CONFIG_DEVFS_FS=y
CONFIG_DEVFS_MOUNT=y
# CONFIG_DEVFS_DEBUG is not set
# CONFIG_DEVPTS_FS is not set
# CONFIG_QNX4FS_FS is not set
# CONFIG_QNX4FS_RW is not set
# CONFIG_ROMFS_FS is not set
CONFIG_EXT2_FS=y
# CONFIG_SYSV_FS is not set
# CONFIG_UDF_FS is not set
# CONFIG_UDF_RW is not set
# CONFIG_UFS_FS is not set
# CONFIG_UFS_FS_WRITE is not set

#
# Network File Systems
#
# CONFIG_CODA_FS is not set
# CONFIG_INTERMEZZO_FS is not set
CONFIG_NFS_FS=y
# CONFIG_NFS_V3 is not set
# CONFIG_NFS_DIRECTIO is not set
CONFIG_ROOT_NFS=y
# CONFIG_NFSD is not set
# CONFIG_NFSD_V3 is not set
# CONFIG_NFSD_TCP is not set
CONFIG_SUNRPC=y
CONFIG_LOCKD=y
# CONFIG_SMB_FS is not set
# CONFIG_NCP_FS is not set
# CONFIG_NCPFS_PACKET_SIGNING is not set
# CONFIG_NCPFS_IOCTL_LOCKING is not set
# CONFIG_NCPFS_STRONG is not set
# CONFIG_NCPFS_NFS_NS is not set
# CONFIG_NCPFS_OS2_NS is not set
# CONFIG_NCPFS_SMALLDOS is not set
# CONFIG_NCPFS_NLS is not set
# CONFIG_NCPFS_EXTRAS is not set
# CONFIG_ZISOFS_FS is not set

#
# Partition Types
#
# CONFIG_PARTITION_ADVANCED is not set
CONFIG_MSDOS_PARTITION=y
# CONFIG_SMB_NLS is not set
# CONFIG_NLS is not set

#
# Console drivers
#
CONFIG_VGA_CONSOLE=y
# CONFIG_VIDEO_SELECT is not set
# CONFIG_MDA_CONSOLE is not set

#
# Frame-buffer support
#
# CONFIG_FB is not set

#
# Sound
#
# CONFIG_SOUND is not set

#
# USB support
#
# CONFIG_USB is not set
```

```
#
# Bluetooth support
#
# CONFIG_BLUEZ is not set

#
# Kernel hacking
#
# CONFIG_DEBUG_KERNEL is not set

#
# Cryptographic options
#
# CONFIG_CRYPTO is not set

#
# Library routines
#
# CONFIG_CRC32 is not set
# CONFIG_ZLIB_INFLATE is not set
# CONFIG_ZLIB_DEFLATE is not set
```

ANEXO 27 Mi Makefile.cfg de svgalib

```
#-----------------------------------------------------------------------
# SVGAlib Compile-time configuration file
#-----------------------------------------------------------------------
# If you change ANYTHING in here you MUST 'make clean' and remake what
# you need.
#
# BEWARE! The toggle settings (INCLUDE_*_DRIVER and such) are set when
# the symbols are set. The value is pointless. Even setting a variable
# to n means yes!

MAJOR_VER = 1
MINOR_VER = 9.18
VERSION = $(MAJOR_VER).$(MINOR_VER)

#-----------------------------------------------------------------------
# Kernel module Configuration Section
#-----------------------------------------------------------------------
# Char major device used by module
SVGALIB_HELPER_MAJOR = 209

# Linux kernel includes
ifeq ($(KERNELRELEASE),)
KERNELRELEASE = $(shell uname -r)
endif

INCLUDEDIR = /lib/modules/$(KERNELRELEASE)/build/include

DIREXIST = $(shell if [ -e $(INCLUDEDIR) ] ; then echo OK ; fi)
ifneq ($(DIREXIST), OK)
INCLUDEDIR = /usr/src/linux/include
endif

ARCH := $(shell uname -m | sed -e s/i.86/i386/ -e s/sun4u/sparc64/ -e s/arm.*/arm/ -e s/sa110/arm/)

#-----------------------------------------------------------------------
# Configuration Section
#-----------------------------------------------------------------------
# Source directory.
#srcdir = /usr/local/src/svgalib-$(VERSION)
srcdir = $(shell sh -c pwd)

confdir = $(srcdir)/src/config

# Common prefix for installation directories.
# NOTE: This directory must exist when you start the install.
TOPDIR=
prefix = $(TOPDIR)/usr/local
exec_prefix = $(prefix)

# Directory where the shared stubs and static library will be installed.
libdir = $(exec_prefix)/lib

# Directory where the shared library will be installed.
sharedlibdir = $(exec_prefix)/lib

# Directory where the font and textmode utilities will be installed.
bindir = $(exec_prefix)/bin

# Directory where the run-time configuration files will be installed.
datadir = $(TOPDIR)/etc/vga

# Directory where the header files will be installed.
includedir = $(prefix)/include

# Directory where the man files will be installed.
mandir = $(prefix)/man
```

```
# Target binary format.
TARGET_FORMAT = elf

# uncomment any of the following line to print some debug messages
# DEBUG = yes
# DEBUG_CONFIG = yes
# DEBUG_ACCEL = yes
# DEBUG_KEYBOARD = yes

# uncomment this if your compiler fails on compiling the assembler in
# src/vgaconvplanar.c, gl/inlstring.h, gl/line.c or gl/scale.c
# NO_ASM = y

# uncomment if you want to set attribute controller and dac without delay
# This breaks original VGA, but seems to work on new cards.
#NO_DELAY = y

# Uncomment this if you want root processes to be able to always get a new
# VC. Alas, some games misuse suid root privs and become root, svgalib cannot
# detect this and will allow Joe blow user to open a new virtual VC. If this
# annoys you, comment out the next line (which is the default already)
ROOT_VC_SHORTCUT = y

# Uncomment thit to use libc's memcpy, instead of simple while loop.
# Use demos/linearspeed to see which is faster to video memory.
# LIBC_MEMCPY = y

# Uncomment this if you want to compile and install the static libs.
# INSTALLSTATICLIB = installstaticlib

# Comment this out if you don't want to install the shared libs.
# If you do not install the shared nor the static libs, 'make static'
# first to enforce just building the static lib, then the demos will
# use this local static library!
INSTALLSHAREDLIB = installsharedlib

# Comment this out if you want to keep old shared images. Old header files,
# library stubs and static libraries CANNOT be kept in public locations
# except when you rename them yourself.
# KEEPSHAREDLIBS = keep

# Comment this out if you don't want to compile and install the utilities.
# INSTALLUTILS = installutils

# Comment this out if you don't want to install the man pages by default
# INSTALLMAN = installman

# Remove the '# ' from one of the next two lines if you want to install the
# man pages compressed (with gzip) or not. If you comment out both lines,
# the Makefiles will try to figure out if your system supports gzipped man
# pages and install them when possible.

# MANFORMAT = compressed
# MANFORMAT = uncompressed

# This is the command to update the man pages whatis database.
# This is a slow command. If you are not very patient, simple
# comment out this line
# MAKEWHATIS = makewhatis # Beware, this will really need a few minutes!

# Comment this out if you use devfs only and don't want to make svga nodes
INSTALLDEV = installdev

#
# Comment out any driver that you don't want included in the library.
#
#INCLUDE_ET4000_DRIVER = y
#INCLUDE_OAK_DRIVER = y
#INCLUDE_EGA_DRIVER = y
```

```
#INCLUDE_MACH32_DRIVER = y
#INCLUDE_ET3000_DRIVER = y
#INCLUDE_GVGA6400_DRIVER = y
#INCLUDE_ATI_DRIVER = y
#INCLUDE_CHIPS_DRIVER = y
#
#INCLUDE_APM_DRIVER = y
#INCLUDE_NV3_DRIVER = y
#INCLUDE_G400_DRIVER = y
#INCLUDE_R128_DRIVER = y
#INCLUDE_VESA_DRIVER = y
#INCLUDE_MX_DRIVER = y
#INCLUDE_RENDITION_DRIVER = y
#INCLUDE_RAGE_DRIVER = y
#INCLUDE_BANSHEE_DRIVER = y
#INCLUDE_SIS_DRIVER = y
#INCLUDE_I740_DRIVER = y
#INCLUDE_I810_DRIVER = y
#INCLUDE_LAGUNA_DRIVER = y
#INCLUDE_TRIDENT_DRIVER = y
#INCLUDE_SAVAGE_DRIVER = y
#INCLUDE_MILLENNIUM_DRIVER = y
#INCLUDE_G450C2_DRIVER = y
#INCLUDE_PM2_DRIVER = y
#
#INCLUDE_NEO_DRIVER = y
#INCLUDE_ET6000_DRIVER = y
#INCLUDE_FBDEV_DRIVER = y
#
#INCLUDE_PARADISE_DRIVER = y
#INCLUDE_ARK_DRIVER = y
#INCLUDE_S3_DRIVER = y
INCLUDE_CIRRUS_DRIVER = y
#INCLUDE_TVGA_DRIVER = y
#INCLUDE_ALI_DRIVER = y
#
#
#
# Comment out any adapter you don't want to autodetect.
#
#INCLUDE_ET4000_DRIVER_TEST = y
INCLUDE_CIRRUS_DRIVER_TEST = y
#INCLUDE_TVGA_DRIVER_TEST = y
#INCLUDE_OAK_DRIVER_TEST = y
#INCLUDE_EGA_DRIVER_TEST = y
#INCLUDE_MACH32_DRIVER_TEST = y
#INCLUDE_GVGA6400_DRIVER_TEST = y
#INCLUDE_S3_DRIVER_TEST = y
#INCLUDE_ET3000_DRIVER_TEST = y
#INCLUDE_ARK_DRIVER_TEST = y
#INCLUDE_ATI_DRIVER_TEST = y
#INCLUDE_ALI_DRIVER_TEST = y
#INCLUDE_CHIPS_DRIVER_TEST = y
#INCLUDE_APM_DRIVER_TEST = y
#INCLUDE_NV3_DRIVER_TEST = y
#INCLUDE_G400_DRIVER_TEST = y
#INCLUDE_R128_DRIVER_TEST = y
#INCLUDE_ET6000_DRIVER_TEST = y
#INCLUDE_MX_DRIVER_TEST = y
#INCLUDE_TRIDENT_DRIVER_TEST = y
#INCLUDE_PARADISE_DRIVER_TEST = y
#INCLUDE_RAGE_DRIVER_TEST = y
#INCLUDE_BANSHEE_DRIVER_TEST = y
#INCLUDE_SIS_DRIVER_TEST = y
#INCLUDE_I740_DRIVER_TEST = y
#INCLUDE_I810_DRIVER_TEST = y
#INCLUDE_LAGUNA_DRIVER_TEST = y
#INCLUDE_NEO_DRIVER_TEST = y
#INCLUDE_SAVAGE_DRIVER_TEST = y
#INCLUDE_MILLENNIUM_DRIVER_TEST = y
```

```
#INCLUDE_RENDITION_DRIVER_TEST = y
#INCLUDE_PM2_DRIVER_TEST = y

#INCLUDE_FBDEV_DRIVER_TEST = y

#Might be too dangerous:
#INCLUDE_VESA_DRIVER_TEST = y

#
# Comment out any dac support that you don't want included in the library.
#
# you must include SIERRA_DAC, if you include any of SCxxxx DACs.


#INCLUDE_NORMAL_DAC = y
#INCLUDE_S3_SDAC_DAC = y
#INCLUDE_S3_GENDAC_DAC = y
#INCLUDE_S3_TRIO64_DAC = y
#INCLUDE_SIERRA_DAC = y
#INCLUDE_SC15025_DAC = y
#INCLUDE_ATT20C490_DAC = y
#INCLUDE_ATT20C498_DAC = y
#INCLUDE_ICW_DAC = y
#INCLUDE_IBMRGB52x_DAC = y
#INCLUDE_SC1148X_DAC = y
#INCLUDE_ICS_GENDAC_DAC = y

#
# Comment out any dac you don't want to autodetect.
# (not all dacs can be autodetected, at this time)
#
#INCLUDE_S3_SDAC_DAC_TEST = y
#INCLUDE_S3_GENDAC_DAC_TEST = y
#INCLUDE_SC15025_DAC_TEST = y
#INCLUDE_ATT20C490_DAC_TEST = y
#INCLUDE_IBMRGB52x_DAC_TEST = y
#INCLUDE_SC1148X_DAC_TEST = y
#INCLUDE_ICS_GENDAC_DAC_TEST = y

# LRMI - disbaled for non i386
LRMI = lrmi

# Location of the svgalib configuration file.
SVGALIB_CONFIG_FILE = $(datadir)/libvga.config

# Defining DYNAMIC enables runtime parsing of the file defined by
# ET4000_REGS (usually /etc/libvga.et4000) for the et4000
# driver. See et4000/README for details. Commenting this out again
# saves binary space.
#
# If you just want to use the et4000.regs in the source directory,
# comment out the definition of DYNAMIC.  DYNAMIC allows development of new
# resolutions without recompiling.
#DYNAMIC = y
#ET4000_REGS = $(datadir)/libvga.et4000

# The EGA driver may load additional modes (SuperEGA cards) like the
# et4000 driver does. Just define the configuration file below.
# [This should be taken with a grain of salt, EGA is untested.]
#EGA_REGS = $(datadir)/libvga.ega

# Defining USE_CLOCKS will cause the ET4000 driver to measure clock
# frequencies (they are not actually used yet).
#USE_CLOCKS = y

# Uncomment to allow mouse type overrides
ALLOW_MOUSE_OVERRIDE = y
#-----------------------------------------------------------------------
# Compiler Section
#-----------------------------------------------------------------------
```

```
# Compiler used.
PC      = ppc386

ifndef CC
        CC     = gcc
endif


ifndef CFLAGS
        OPTIMIZE = -fomit-frame-pointer -O2 -fno-strength-reduce -pipe -g
else
        OPTIMIZE := $(CFLAGS)
endif

# You might want to add -m386 here if you have a recently installed
# (486 configured) compiler on a 386. The only real difference is the
# generous alignment padding of function entry-points for the 486.
WARN     = -Wall -Wstrict-prototypes
INCLUDES = -I$(srcdir)/include -I.
CFLAGS   = $(WARN) $(DLLFLAGS) $(INCLUDES) $(OPTIMIZE) $(DEFINES)
LDFLAGS   = -s

# additional flags for shared lib.
DLLFLAGS = -fPIC

# Utilites used.
AR       = ar
INSTALL_PROGRAM       = install -c -s -m 755 -o root -g bin
INSTALL_SHLIB  = install -c -m 755 -o root -g bin
INSTALL_DATA   = install -c -m 644 -o root -g bin

ifneq ($(ARCH),i386)
        NO_ASM = y
        INCLUDE_VESA_DRIVER=

        INCLUDE_CHIPS_DRIVER=
        INCLUDE_ET4000_DRIVER=
        INCLUDE_MACH32_DRIVER=
        INCLUDE_NEO_DRIVER=
        INCLUDE_TVGA_DRIVER=
        INCLUDE_I810_DRIVER=
        LRMI=
endif

ifeq ($(ARCH),ppc)
        INCLUDE_S3_DRIVER=
endif

ifneq ($(INCLUDE_S3_DRIVER),y)
        INCLUDE_NORMAL_DAC =
        INCLUDE_S3_SDAC_DAC =
        INCLUDE_S3_GENDAC_DAC =
        INCLUDE_S3_TRIO64_DAC =
        INCLUDE_SIERRA_DAC =
        INCLUDE_SC15025_DAC =
        INCLUDE_ATT20C490_DAC =
        INCLUDE_ATT20C498_DAC =
        INCLUDE_ICW_DAC =
        INCLUDE_IBMRGB52x_DAC =
        INCLUDE_SC1148X_DAC =
endif
```

ANEXO 28 Mi .config de uClibC

```
#
# Automatically generated make config: don't edit
#

#
# Target Architecture Features and Options
#
HAVE_ELF=y
# CONFIG_GENERIC_386 is not set
CONFIG_386=y
# CONFIG_486 is not set
# CONFIG_586 is not set
# CONFIG_586MMX is not set
# CONFIG_686 is not set
# CONFIG_PENTIUMIII is not set
# CONFIG_PENTIUM4 is not set
# CONFIG_K6 is not set
# CONFIG_K7 is not set
# CONFIG_CRUSOE is not set
# CONFIG_WINCHIPC6 is not set
# CONFIG_WINCHIP2 is not set
# CONFIG_CYRIXIII is not set
UCLIBC_HAS_MMU=y
UCLIBC_HAS_FLOATS=y
HAS_FPU=y
# DO_C99_MATH is not set
WARNINGS="-Wall"
KERNEL_SOURCE="/home/bootdisk/linux"
C_SYMBOL_PREFIX=""
HAVE_DOT_CONFIG=y

#
# General Library Settings
#
DOPIC=y
HAVE_SHARED=y
# ADD_LIBGCC_FUNCTIONS is not set
BUILD_UCLIBC_LDSO=y
LDSO_LDD_SUPPORT=y
UCLIBC_CTOR_DTOR=y
UCLIBC_HAS_THREADS=y
UCLIBC_HAS_LFS=y
# MALLOC is not set
MALLOC_930716=y
UCLIBC_DYNAMIC_ATEXIT=y
HAS_SHADOW=y
UCLIBC_HAS_REGEX=y
UNIX98PTY_ONLY=y
ASSUME_DEVPTS=y

#
# Networking Support
#
# UCLIBC_HAS_IPV6 is not set
# UCLIBC_HAS_RPC is not set

#
# String and Stdio Support
#
# UCLIBC_HAS_WCHAR is not set
# USE_OLD_VFPRINTF is not set

#
# Library Installation Options
#
SHARED_LIB_LOADER_PATH="/lib"
```

```
DEVEL_PREFIX="/usr/$(TARGET_ARCH)-linux-uclibc"
SYSTEM_DEVEL_PREFIX="$(DEVEL_PREFIX)"
DEVEL_TOOL_PREFIX="$(DEVEL_PREFIX)/usr"

#
# uClibc hacking options
#
# DODEBUG is not set
# DOASSERTS is not set
# SUPPORT_LD_DEBUG is not set
# SUPPORT_LD_DEBUG_EARLY is not set
```

ANEXO 29 Mi Config.h de busybox

```
/* vi: set sw=4 ts=4: */
// This file defines the feature set to be compiled into busybox.
// When you turn things off here, they won't be compiled in at all.
//
//// This file is parsed by sed.  You MUST use single line comments.
//   i.e.,  //#define BB_BLAH
//
//
// BusyBox Applications
//#define BB_ADJTIMEX
//#define BB_AR
#define BB_ASH
//#define BB_BASENAME
//#define BB_CAT
//define BB_CHGRP
//#define BB_CHMOD
//#define BB_CHOWN
//#define BB_CHROOT
//#define BB_CHVT
//#define BB_CLEAR
//#define BB_CMP
//#define BB_CP
//#define BB_CPIO
//#define BB_CUT
#define BB_DATE
//#define BB_DC
//#define BB_DD
//#define BB_DEALLOCVT
//#define BB_DF
//#define BB_DIRNAME
//#define BB_DMESG
//#define BB_DOS2UNIX
//#define BB_DPKG
//#define BB_DPKG_DEB
//#define BB_DUTMP
//#define BB_DU
//#define BB_DUMPKMAP
#define BB_ECHO
//#define BB_ENV
//#define BB_EXPR
//#define BB_FBSET
//#define BB_FDFLUSH
//#define BB_FIND
//#define BB_FREE
//#define BB_FREERAMDISK
//#define BB_FSCK_MINIX
//#define BB_GETOPT
//#define BB_GREP
//#define BB_GUNZIP
//#define BB_GZIP
//#define BB_HALT
//#define BB_HEAD
//#define BB_HOSTID
//#define BB_HOSTNAME
//#define BB_HUSH
//#define BB_ID
#define BB_IFCONFIG
#define BB_INIT
#define BB_INSMOD
//#define BB_KILL
//#define BB_KILLALL
//#define BB_KLOGD
//#define BB_LASH
//#define BB_LENGTH
//#define BB_LN
//#define BB_LOADACM
```

```
//#define BB_LOADFONT
//#define BB_LOADKMAP
//#define BB_LOGGER
//#define BB_LOGNAME
//#define BB_LOSETUP
#define BB_LS
//#define BB_LSMOD
//#define BB_MAKEDEVS
//#define BB_MD5SUM
//#define BB_MKDIR
//#define BB_MKFIFO
//#define BB_MKFS_MINIX
//#define BB_MKNOD
//#define BB_MKSWAP
//#define BB_MKTEMP
//#define BB_MODPROBE
//#define BB_MORE
//#define BB_MOUNT
//#define BB_MSH
//#define BB_MT
//#define BB_MV
//#define BB_NC
//#define BB_NSLOOKUP
//#define BB_PIDOF
#define BB_PING
//#define BB_PIVOT_ROOT
//#define BB_POWEROFF
//#define BB_PRINTF
//#define BB_PS
//#define BB_PWD
//#define BB_RDATE
//#define BB_READLINK
//#define BB_REBOOT
//#define BB_RENICE
//#define BB_RESET
//#define BB_RM
//#define BB_RMDIR
//#define BB_RMMOD
//#define BB_ROUTE
//#define BB_RPM2CPIO
//#define BB_SED
//#define BB_SETKEYCODES
//#define BB_SLEEP
//#define BB_SORT
//#define BB_STTY
//#define BB_SWAPONOFF
//#define BB_SYNC
//#define BB_SYSLOGD
//#define BB_TAIL
//#define BB_TAR
//#define BB_TEE
//#define BB_TEST
//#define BB_TELNET
//#define BB_TFTP
//#define BB_TIME
//#define BB_TOP
//#define BB_TOUCH
//#define BB_TR
//#define BB_TRACEROUTE
//#define BB_TRUE_FALSE
#define BB_TTY
//#define BB_UNIX2DOS
//#define BB_UUENCODE
//#define BB_UUDECODE
//#define BB_UMOUNT
//#define BB_UNIQ
#define BB_UNAME
//#define BB_UPDATE
//#define BB_UPTIME
//#define BB_USLEEP
```

```
//#define BB_VI
//#define BB_WATCHDOG
//#define BB_WC
//#define BB_WGET
//#define BB_WHICH
//#define BB_WHOAMI
//#define BB_XARGS
//#define BB_YES
// End of Applications List
//
//
//
// -------------------------------------------------------
// This is where feature definitions go.  Generally speaking,
// turning this stuff off makes things a bit smaller (and less
// pretty/useful).
//
//
// If you enabled one or more of the shells, you may select which one
// should be run when sh is invoked:
#define BB_FEATURE_SH_IS_ASH
//#define BB_FEATURE_SH_IS_HUSH
//#define BB_FEATURE_SH_IS_LASH
//#define BB_FEATURE_SH_IS_MSH
//
// BusyBox will, by default, malloc space for its buffers.  This costs code
// size for the call to xmalloc.  You can use the following feature to have
// them put on the stack.  For some very small machines with limited stack
// space, this can be deadly.  For most folks, this works just fine...
//#define BB_FEATURE_BUFFERS_GO_ON_STACK
// The third alternative for buffer allocation is to use BSS.  This works
// beautifully for computers with a real MMU (and OS support), but wastes
// runtime RAM for uCLinux.  This behavior was the only one available for
// BusyBox versions 0.48 and earlier.
//#define BB_FEATURE_BUFFERS_GO_IN_BSS
//
// Turn this on to use Erik's very cool devps, and devmtab kernel drivers,
// thereby eliminating the need for the /proc filesystem and thereby saving
// lots and lots memory for more important things.  NOTE:  If you enable this
// feature, you _must_ have patched the kernel to include the devps patch that
// is included in the busybox/kernel-patches directory.  You will also need to
// create some device special files in /dev on your embedded system:
//        mknod /dev/mtab c 10 22
//        mknod /dev/ps c 10 21
// I emailed Linus and this patch will not be going into the stock kernel.
//#define BB_FEATURE_USE_DEVPS_PATCH
//
// show verbose usage messages
//#define BB_FEATURE_VERBOSE_USAGE
//
// Use termios to manipulate the screen ('more' is prettier with this on)
//#define BB_FEATURE_USE_TERMIOS
//
// calculate terminal & column widths (for more, ls, and telnet)
//#define BB_FEATURE_AUTOWIDTH
//
// show username/groupnames for ls
//#define BB_FEATURE_LS_USERNAME
//
// show file timestamps in ls
//#define BB_FEATURE_LS_TIMESTAMPS
//
// enable ls -p and -F
//#define BB_FEATURE_LS_FILETYPES
//
// sort the file names
//#define BB_FEATURE_LS_SORTFILES
//
// enable ls -R
//#define BB_FEATURE_LS_RECURSIVE
```

```
//
// enable ls -L
//#define BB_FEATURE_LS_FOLLOWLINKS
//
// Use color to identify different file types
//#define BB_FEATURE_LS_COLOR
//
// Disable for a smaller (but less functional) ping
//#define BB_FEATURE_FANCY_PING
//
// Make init use a simplified /etc/inittab file (recommended).
#define BB_FEATURE_USE_INITTAB
//
//Enable init being called as /linuxrc
#define BB_FEATURE_LINUXRC
//
//Have init enable core dumping for child processes (for debugging only)
//#define BB_FEATURE_INIT_COREDUMPS
//
//Make sure nothing is printed to the console on boot
//#define BB_FEATURE_EXTRA_QUIET
//
// enable syslogd -R remotehost
//#define BB_FEATURE_REMOTE_LOG
//
// enable syslogd -C
//#define BB_FEATURE_IPC_SYSLOG
//
//Disable for a simple tail implementation (2.34k vs 3k for the full one).
//Both provide 'tail -f', but this cuts out -c, -q, -s, and -v.
//#define BB_FEATURE_FANCY_TAIL
//
// Enable support for loop devices in mount
//#define BB_FEATURE_MOUNT_LOOP
//
// Enable support for a real /etc/mtab file instead of /proc/mounts
//#define BB_FEATURE_MTAB_SUPPORT
//
// Enable support for mounting remote NFS volumes.
// You may need to mount with "-o nolock" if you are
// not running a local portmapper daemon...
//
// If you are using uClibc, be sure that you've already compiled
// uClibc with INCLUDE_RPC=true (contained in the Config file)
//#define BB_FEATURE_NFSMOUNT
//
// Enable support forced filesystem unmounting
// (i.e., in case of an unreachable NFS system).
//#define BB_FEATURE_MOUNT_FORCE
//
// Enable support for creation of tar files.
//#define BB_FEATURE_TAR_CREATE
//
// Enable support for "--exclude" and "-X" for excluding files
//#define BB_FEATURE_TAR_EXCLUDE
//
// Enable support for tar -z option (currently only works for inflating)
//#define BB_FEATURE_TAR_GZIP
//
// Enable reverse sort
//#define BB_FEATURE_SORT_REVERSE
//
// Enable uniqe sort
//#define BB_FEATURE_SORT_UNIQUE
//
// Enable command line editing in the shell.
// Only relevant if a shell is enabled. On by default.
//#define BB_FEATURE_COMMAND_EDITING
//
// Enable tab completion in the shell.  This is now working quite nicely.
```

```
// This feature adds a bit over 4k. Only relevant if a shell is enabled.
//#define BB_FEATURE_COMMAND_TAB_COMPLETION
//
// Attempts to match usernames in a ~-prefixed path
//#define BB_FEATURE_COMMAND_USERNAME_COMPLETION
//
//Allow the shell to invoke all the compiled in BusyBox applets as if they
//were shell builtins.  Nice for staticly linking an emergency rescue shell,
//among other things. Off by default.
// Only relevant if a shell is enabled.
//#define BB_FEATURE_SH_STANDALONE_SHELL
//
//When this is enabled, busybox shell applets can be called using full path
//names.  This causes applets (i.e., most busybox commands) to override
//real commands on the filesystem.  For example, if you run run /bin/cat, it
//will use BusyBox cat even if /bin/cat exists on the filesystem and is _not_
//busybox.  Some systems want this, others do not.  Choose wisely.  :-) This
//only has meaning when BB_FEATURE_SH_STANDALONE_SHELL is enabled.
// Only relevant if a shell is enabled. Off by default.
//#define BB_FEATURE_SH_APPLETS_ALWAYS_WIN
//
// Uncomment this option for a fancy shell prompt that includes the
// current username and hostname.  On systems that don't have usernames
// or hostnames, this can look hideous.
// Only relevant if a shell is enabled.
//#define BB_FEATURE_SH_FANCY_PROMPT
//
// Uncomment this option to disable job control.  Job control lets you
// run jobs in the background (which completely useless for is all you
// are doing is running scripts).  Disabing this is bad for interactive
// use, since when you hit ^C in an application, it will also kill the
// shell.  This adds about 2.5k on an x86 system.
//#define BB_FEATURE_ASH_JOB_CONTROL
//
//Turn on extra fbset options
//#define BB_FEATURE_FBSET_FANCY
//
//Turn on fbset readmode support
//#define BB_FEATURE_FBSET_READMODE
//
// Support insmod/lsmod/rmmod for post 2.1 kernels
#define BB_FEATURE_NEW_MODULE_INTERFACE
//
// Support insmod/lsmod/rmmod for pre 2.1 kernels
//#define BB_FEATURE_OLD_MODULE_INTERFACE
//
// Support module version checking
//#define BB_FEATURE_INSMOD_VERSION_CHECKING
//
// Support for uClinux memory usage optimization, which will load the image
// directly into the kernel memory.  This divides memory requrements by three.
// If you are not running uClinux (i.e., your CPU has an MMU) leave this
// disabled...
//#define BB_FEATURE_INSMOD_LOADINKMEM
//
// Support for Minix filesystem, version 2
//#define BB_FEATURE_MINIX2
//
// Enable ifconfig status reporting output -- this feature adds 7k.
#define BB_FEATURE_IFCONFIG_STATUS
//
// Enable ifconfig slip-specific options "keepalive" and "outfill"
//#define BB_FEATURE_IFCONFIG_SLIP
//
// Enable ifconfig options "mem_start", "io_addr", and "irq".
//#define BB_FEATURE_IFCONFIG_MEMSTART_IOADDR_IRQ
//
// Enable ifconfig option "hw".  Currently works for only with "ether".
//#define BB_FEATURE_IFCONFIG_HW
//
```

```
// Allows "broadcast +" to set broadcast automatically based on hostaddr
// and netmask, at a cost of about 100 bytes of code (i386).
//#define BB_FEATURE_IFCONFIG_BROADCAST_PLUS
//
// Enable busybox --install [-s]
// to create links (or symlinks) for all the commands that are
// compiled into the binary.  (needs /proc filesystem)
//#define BB_FEATURE_INSTALLER
//
// Enable a nifty progress meter in wget (adds just under 2k)
//#define BB_FEATURE_WGET_STATUSBAR
//
// Enable HTTP authentication in wget
//#define BB_FEATURE_WGET_AUTHENTICATION
//
// Clean up all memory before exiting -- usually not needed
// as the OS can clean up...  Don't enable this unless you
// have a really good reason for cleaning things up manually.
//#define BB_FEATURE_CLEAN_UP
//
// Support for human readable output by ls, du, etc.(example 13k, 23M, 235G)
//#define BB_FEATURE_HUMAN_READABLE
//
// Support for the find -type option.
//#define BB_FEATURE_FIND_TYPE
//
// Support for the find -perm option.
//#define BB_FEATURE_FIND_PERM
//
// Support for the find -mtime option.
//#define BB_FEATURE_FIND_MTIME
//
//// Support for the find -newer option.
//#define BB_FEATURE_FIND_NEWER
//
// Support for the -A -B and -C context flags in grep
//#define BB_FEATURE_GREP_CONTEXT
//
// Support for the EGREP applet (alias to the grep applet)
//#define BB_FEATURE_GREP_EGREP_ALIAS
//
// Tell tftp what commands that should be supported.
//#define BB_FEATURE_TFTP_PUT
//#define BB_FEATURE_TFTP_GET
//
// features for vi
//#define BB_FEATURE_VI_COLON        // ":" colon commands, no "ex" mode
//#define BB_FEATURE_VI_YANKMARK            // Yank/Put commands and Mark cmds
//#define BB_FEATURE_VI_SEARCH              // search and replace cmds
//#define BB_FEATURE_VI_USE_SIGNALS  // catch signals
//#define BB_FEATURE_VI_DOT_CMD             // remember previous cmd and "." cmd
//#define BB_FEATURE_VI_READONLY            // vi -R and "view" mode
//#define BB_FEATURE_VI_SETOPTS             // set-able options,  ai ic showmatch
//#define BB_FEATURE_VI_SET          // :set
//#define BB_FEATURE_VI_WIN_RESIZE   // handle window resize
//
// Enable a if you system have setuped locale
//#define BB_LOCALE_SUPPORT
//
// Support for TELNET to pass TERM type to remote host.  Adds 384 bytes.
//#define BB_FEATURE_TELNET_TTYPE
//
// Support for devfs.
#define BB_FEATURE_DEVFS
//
// End of Features List
//
//
//
//
```

```
//
//
//----------------------------------------------------
// Nothing beyond this point should ever be touched by
// mere mortals so leave this stuff alone.
//
#include <features.h>
#if defined(__uClinux__)
        #undef BB_ASH                 /* Not even a chance it will work */
        #undef BB_RPM2CPIO            /* Uses gz_open(), which uses fork() */
        #undef BB_DPKG_DEB            /* Uses gz_open(), which uses fork() */
        #undef BB_FEATURE_TAR_GZIP    /* Uses fork() */
        #undef BB_UPDATE              /* Uses daemon() */
#endif
#if defined BB_ASH || defined BB_HUSH || defined BB_LASH || defined BB_MSH
        #if defined BB_FEATURE_COMMAND_EDITING
                #define BB_CMDEDIT
        #else
                #undef BB_FEATURE_COMMAND_EDITING
                #undef BB_FEATURE_COMMAND_TAB_COMPLETION
                #undef BB_FEATURE_COMMAND_USERNAME_COMPLETION
                #undef BB_FEATURE_SH_FANCY_PROMPT
        #endif
#else
        #undef BB_FEATURE_SH_APPLETS_ALWAYS_WIN
        #undef BB_FEATURE_SH_STANDALONE_SHELL
        #undef BB_FEATURE_SH_FANCY_PROMPT
#endif
//
#if (defined BB_ASH || defined BB_HUSH || defined BB_MSH) && ! defined BB_TEST
        #define BB_TEST
#endif
//
#ifdef BB_KILLALL
        #ifndef BB_KILL
                #define BB_KILL
        #endif
#endif
//
#ifndef BB_INIT
        #undef BB_FEATURE_LINUXRC
#endif
//
#if defined BB_MOUNT && defined BB_FEATURE_NFSMOUNT
        #define BB_NFSMOUNT
#endif
//
#if defined BB_FEATURE_AUTOWIDTH
        #ifndef BB_FEATURE_USE_TERMIOS
                #define BB_FEATURE_USE_TERMIOS
        #endif
#endif
//
#if defined BB_INSMOD || defined BB_LSMOD
        #if ! defined BB_FEATURE_NEW_MODULE_INTERFACE && ! defined BB_FEATURE_OLD_MODULE_INTERFACE
                #define BB_FEATURE_NEW_MODULE_INTERFACE
        #endif
#endif
//
#ifdef BB_UNIX2DOS
        #define BB_DOS2UNIX
#endif
//
#ifdef BB_SYSLOGD
        #if defined BB_FEATURE_IPC_SYSLOG
                #define BB_LOGREAD
        #endif
#endif
//
#if defined BB_ASH && defined BB_FEATURE_SH_IS_ASH
```

```
# define shell_main ash_main
#elif defined BB_HUSH && defined BB_FEATURE_SH_IS_HUSH
# define shell_main hush_main
#elif defined BB_LASH && defined BB_FEATURE_SH_IS_LASH
# define shell_main lash_main
#elif defined BB_MSH && defined BB_FEATURE_SH_IS_MSH
# define shell_main msh_main
#endif
```

ANEXO 30 Mis scripts para trabajar mas comodo

Crear RAMDISK y volcar todo el contenido del RFS en una imagen:

```
#!/bin/bash
dd if=/dev/zero of=rootfs bs=1k count=700
mke2fs -F -m 0 -N 400 rootfs
mount rootfs -o loop loop
rmdir loop/lost+found
cp -a rfloppy/* loop
chown -R 0:0 loop/*
umount loop
dd if=rootfs bs=1k | gzip -v9 > rootfs.gz
sh ./btd.sh

(Guardarlo como crea.sh por ejemplo)
```

Creador de diskette con Kernel+RFS:

```
#!/bin/bash
dd if=bzImage_NFSv1 of=/dev/fd0 bs=1k
rdev /dev/fd0 /dev/fd0
rdev -R /dev/fd0 0
# Recorda que este es el numero magico, en este caso '16930'
# Resultante de sumar 16384+546
rdev -r /dev/fd0 16930
# Recorda que el numero '546' en este caso es el tamaño que te proporciono
# dd cuando le sumaste el cociente (1) al tamaño del kernel
dd if=rootfs.gz of=/dev/fd0 bs=1k seek=546

(Guardarlo como btd.sh por ejemplo)
```

ANEXO 31 *Troubleshooting...*

Este anexo es el resultado de largo tiempo de resolucion de problemas, apuntes, documentaciones oficiales consultadas, entre otras fuentes.

DHCP

Diagnostico, solucion de problemas (I)

Para verificar si dhcpd esta corriendo en nuestro server:
`netstat -apn | grep ":67 "`

Deberia darnos una salida como esta:
```
udp      0    0 0.0.0.0:67            0.0.0.0:*     5958/dhcpd
```

Si dhcp no queda en modo 'start' consistentemente despues del aranque del server:
`# chkconfig –level 3 5 dhcpd on`
Esto hara que dhcp corra en los runlevels 3 & 5 apenas arranca el server.

Diagnostico, solucion de problemas (II)

Debemos controlar que la linea '**option root-path "192.168.0.254:/opt/directorio";**' apunte a donde esta el root filesystem. El error tipico es ver como se genera un 'KERNEL PANIC' porque el kernel no sabe donde esta su root filesystem.
Cuando compartimos en un mismo switch 2 o mas servidores dhcp, incluimos en la 1er linea:

`authoritative;` # NO SIEMPRE FUNCIONA!

TFTP

“File Not Found”
Esto es común, Sucede cuando el cliente trata de traer el kernel.
Verificar en dhcpd.conf que la ruta hacia el kernel, inclusive el kernel es el correcto.
Recordar que: `filename "/lts/2.4.26-ltsp-3/pxelinux.0";`
contiene la ruta completa: `/tftpboot//lts/2.4.26-ltsp-3/pxelinux.0`

Verificando tftp:
`# netstat -apn | grep ":69 "`
Deberia mostrarnos:
```
udp      0    0 0.0.0.0:69            0.0.0.0:*    6122/in.tftpd
```
Parada & arranque:
`# service xinetd [start | stop | status | restart]`

Troubleshooting NFS, diagnostico:
Para comprobar el funcionamiento de nfs en nuestro server:

`# showmount -e`

Deberiamos obtener algo parecido a esto:

```
Export list for server.ltsp:
/opt/ltsp/i386          192.168.0.0/255.255.255.0
/var/opt/ltsp/swapfiles 192.168.0.0/255.255.255.0
```

Para manipular el servicio:

```
# service nfs [start | stop | status | restart]
```

Terminamos reiniciando el servicio xinetd tambien: service xinetd restart

"nfs warning: server not responding..."

Esto sucede porque los kernels 2.6.x usan 32kb para el tamaño de los paquetes nfs, mientras que los kernels 2.4.x usan bloques de 8kb.
Particularmente lo sufren las placas de red 10BaseT GENERICAS (y/o baratas) con un cliente con kernel 2.4 y un server con kernel 2.6.
Para mantener compatibilidad, el kernel 2.4 trata de juntar datagramas de 1500 bytes hasta completar 32kb, sucediento gralmente un time-out en el cliente NFS antes de poder traer todos los datagramas para completar un bloque de 32kb.

Agregar a nuestro dhcpd.conf:

```
option option-128 code 128 = string;
option option-129 code 129 = text;
option option-128  e4:45:74:68:00:00;
option option-129 "MOPTS=nolock,ro,wsize=2048,rsize=2048";

host ws001 {
          hardware ethernet    00:e0:7d:9d:ef:19;
          fixed-address        192.168.0.1;
          filename             "/lts/vmlinuz-2.4.26-ltsp-3";
          option option-128  e4:45:74:68:00:00;
          option option-129 "MOPTS=nolock,ro,wsize=2048,rsize=2048";
                        }
```

NFS a la hora de montar mi RootFileSystem, emite codigo de error "101"

```
Looking up port of RPC 100003/2 on 192.168.16.65
RPC: sendmsg returned error 101
portmap: RPC call returned error 101
Root-NFS: Unable to get nfsd port number from server, using default
Looking up port of RPC 100005/1 on 192.168.16.65
RPC: sendmsg returned error 101
portmap: RPC call returned error 101
Root-NFS: Unable to get mountd port number from server, using default
RPC: sendmsg returned error 101
mount: RPC call returned error 101
Root-NFS: Server returned error -101 while mounting /home/suresh/iso/bsp-sharp7-
1.2.9/nfsroot
VFS: Unable to mount root fs via NFS, trying floppy.
VFS: Cannot open root device "nfs" or unknown-block(2,0)
Please append a correct "root=" boot option
Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(2,0)
```

A no asustarse!

Simplemente has omitido la dirección IP de donde el Kernel debe extraer el RootFileSystem, o has escrito mal la dirección IP del recurso NFS donde se encuentra tu RootFileSystem.
Checa /etc/dhcp.conf y corrigelo.

XDMCP

“Solo veo una X sobre un fondo gris”
Lo clasico es ver en el cliente una pantalla gris con una “X” como puntero del raton.
Esto significa que X esta corriendo localmente poero no puede contactar con el Display Manager remoto. De todas formas, siguiendo los pasos del ANEXO 9 NO DEBERIAS TENER NINGUN INCONVENIENTE en configurar y correr correctamente XDMCP.

Verificar si XDMCP esta disponible:
Para verificar si tu DM (Display Manager) esta escuchando pedidos XDMCP , usamos el siguiente comando:

```
# netstat -apn | grep ":177 "
```

Deberias ver una linea igual a esta:

```
udp      0    0 0.0.0.0:177           0.0.0.0:*     5958/gdm
```

Para reiniciar XDMCP en caliente sige la secuencia:

```
# init 3
# init 5
```

Debuggeando “SYSLOG”

Cuando usamos LTSP para los clientes, suele suceder que al final del boot de nuestra maquina aparezca un error relacionado al demonio SYSLOG, ya que los clientes utilizan el metodo de logs remotos con el servidor...

Modificar /etc/sysconfig/syslog

```
SYSLOGD_OPTIONS="-m 0"
```

por

```
SYSLOGD_OPTIONS="-m 0 -r"
```

Si no modificamos esto, nos dara un error al final del booteo del cliente, este error avisa que el demonio syslog no puede funcionar de forma remota.
Esto es para que sea posible la generacion remota de logs.
Syslogd de esta forma escucha a la red. Por defecto syslog esta en modo “standalone” o sea local.

Floppy imposible de leer

Diskette Etherboot: Despues de arrancar la maquina la pantalla se llena de lineas como:
0200
0cfe

0000
0000
Significa que el diskette esta en mal estado. Reeamplazarlo.

Error Etherboot F?F?F?

Etherboot arranca, imprime "Searching for DHCP... F?F?F?”
Significa que Etherboot recibe ofertas DHCP con el campo "filename" incorrecto.
Es por eso que etherboot no puede determinar cual kernel cargar.
Una entrada tipica podria ser:
option filename "/lts/vmlinuz-2.4.26-ltsp-2";

Etherboot descarga el Kernel, pero imprime los puntos muy lentamente (cada nn seg)

Desabilita el firewall de tu servidor y verifica que tftpd este corriendo con el comando `netstat`:

```
# netstat -lnp | grep ":69"
```

Verifica que el puerto 69 (protocolo udp) este habilitado.

Problemas en la resolucion de pantalla (aka ¡Monitores Samsung SyncMater 3-3Ne no funcionan con LTSP!)

LTSP tiene por defecto 3 resoluciones: 1024*768, 800*600 y 640*480 pixels.
Algunos monitores no soportan las tasas de refresco standard de Xorg.
Solucion rapida, lts.conf:

```
X_MODE_0            = 800x600
X_HORZSYNC          = 28-49
Y_VERTREFRESH       = 43-72
```

Esto funciona en el 90% de los casos.
Solución para expertos (creación de Modelines para un monitor determinado):
`http://koala.ilog.fr/cgi-bin/nph-colas-modelines` (frances)
`http://xtiming.sourceforge.net/cgi-bin/xtiming.pl` (ingles)

¿Renegando con la rueda del Mouse?

Para la mayoria de los ratones con rueda, agregando esta linea a `/opt/ltsp/i386/etc/lts.conf` solucionara el problema:

Para Mouses PS/2
```
X_MOUSE_PROTOCOL = "IMPS/2"
```

Ejemplo practico:

```
[ws001]
        X_MOUSE_PROTOCOL = "IMPS/2"
```

Para Mouses Seriales:

```
[ws001]
        X_MOUSE_DEVICE   = "/dev/ttyS0"
        X_MOUSE_PROTOCOL = "Intellimouse"
```

El teclado se dispara! repite mas de 50 caracteres/seg!

Basicamente el problema lo tienen las placas PC-CHIPS viejas, el problema radica en que de alguna forma el control de energia (ACPI) de estas placas afecta al kernel, para solucionarlo:

archivo dhcpd.conf:

```
option option-128 e4:45:74:68:00:00;
option option-129 "acpi=off";
```

¡La terminal se freeza cuando navego por internet con firefox o uso openoffice!

Esto sucede porque la terminal no tiene suficiente memoria para poder manejar estas aplicaciones. Simplemente agregue memoria swap al cliente en lts.conf, un valor de 96m es recomendable en maquinas con 16MB ram.

```
SWAPFILE_SIZE            = 96m
```

La terminal se freeza aleatoriamente

Si constante y aleatoriamente se freeza la maquina cliente, cuando visitas paginas con contenido de Adobe Flash con las placas de video i810, la solucion no es ni agregarle mas memoria a la maquina, ni agregarle mas memoria swap, solo incrementa la cantidad de memoria RAM de video:

```
X_VIDEORAM     = 8192
```

Mensaje “syslogd: cannot write to remote file handle on 192.168.0.254:514”

Esto se debe a que como se explico anteriormente, syslog por defecto no acepta llamadas remotas. Solucion:

Ubuntu

Debes editar el script /etc/init.d/sysklogd y cerca de la linea 15, deberias ver una linea que dice:

```
SYSLOGD="-u syslog"
```

Necesitas agregarle el parametro '-r',

Ahora quedaria algo asi:

```
SYSLOGD="-r -u syslog"
```

Redhat RHEL3, familia y posteriores
Deberias encontrar un archivo llamado /etc/sysconfig/syslog. Necesitas editarlo y encontrar la linea que dice:

```
SYSLOGD_OPTIONS="-m 0"
```

y agregarle '-r', quedaria asi:

```
SYSLOGD_OPTIONS="-m 0 -r"
```

SuSE 10.0 y posteriores
Descomenta en /etc/syslog-ng/syslog-ng.conf en la linea #42. Opcionalemente modifica el valor ("0.0.0.0") del parametro de tu subred. Otra opcion, como explique antes, es desinstalar syslog-ng e instalar 'syslog' con Yast.

Kernel panic cuando trata de detectar una placa de red ISA!

Esto sucede porque el kernel no tiene posibilidad de detectar este tipo de dispositivos ISA.

Solucion:
Simplemente vaya a su archivo dhcpd.conf y agregue antes del host pertinente las siguientes lineas, si es que no estuviese declarado antes en el archivo:

```
option option-128 code 128 = string;
option option-129 code 129 = text
```

Luego en el host agregue:

```
option option-128     e4:45:74:68:00:00;
option option-129     "NIC=ne IO=0x300";
```

La opcion `option-128  e4:45:74:68:00:00`, es un NUMERO. MAGICO que le dice a etherboot que las siguientes opciones son validas y deben ser respetadas.
¡¡¡NO ES UNA MAC ADDRESS!!!

La opcion `option-129  "NIC=ne IO=0x300"` se lee de la siguiente manera: NIC indica el nombre del modulo (ne=ne2000), IO indica el puerto de la placa y el numero de interrupcion. No son valores arbitrarios, ya estan pre-establecidos para cada placa de este tipo.
NO OLVIDARSE DE REINICIAR DHCP!

Mensaje “Creating new ramdisk to hold our root fs...”

```
Mounting root filesystem: /opt/ltsp/i386 from: [tu_IP]
```

```
mount: RPC: timed out
mount: nfsmount failed: Bad file descriptor
mount:Mounting [tu_IP]:/opt/ltsp/i386 on /newroot/nfsroot failed:
Invalid argument
```

```
Error! Failed to mount the root directory via NFS!
Posible reasons include:
1) NFS services may not be running on the server
2) Workstation IP does not map a hostname, either in /etc/hosts,
or in DNS
3)Wrong address for NFS server in the DHCP config file
4)Wrong pathname for root directory in the dhcp config file
```

```
Kernel panic - not syncing: Attempted to kill init!
```

Es porque no tenemos el nombre de host (nuestro server) y el dominio identicos a `/etc/ltsp/dhcpd.conf` y `/etc/hosts`

Mensaje “make: *** [init/main.o] Error 1”

Cuando trato de compilar mi Kernel 2.4.* tengo un error parecido a este:

```
/usr/src/linux/include/asm/unistd.h:375: warning: conflicting
types for
built-in function '_exit'
make: *** [init/main.o] Error 1"
```

Esto sucede porque estas tratando de compilar un Kernel de la seri 2.4.* con gcc version 4.*, instala si es posible una gcc 3.*.* o como dije al principio del libro, agenciate de una maquina para uso exclusivo de la compilacion con un SO viejo, como Mandrake 9 o 10 o Red Hat 7, 8 o 9.

Si te animas a instalar una gcc mas vieja, luego deberas indicarle al sistema cual debera usar, ya que si no le dices, seguira usando la gcc 4.*, para ellos solo haz lo siguiente:

```
# rm /usr/bin/gcc
# ln -s /usr/bin/gcc3.* /usr/bin/gcc
```

Códigos de error de LiLo

Estos son los códigos de error que presenta LiLo, te lo dejo porque si utilizas EtherBoot con disco duro, quizas experimentes algun error.

Código	Problema
(nada)	Ninguna parte de LiLo ha sido cargada. LiLo no ha sido instalado en el disco duro, o no has hecho activa la particion de arranque del disco
L	La primera etapa del cargador de arranque (booetador) ha sido cargada y arrancó, pero no puede levantar la segunda etapa del cargador. Los dos digitos del codigo de error indican el tipo de problema. Esta condicion indica generalmente la falla en la geometria del disco, o sea, los parametros del mismo no son los correctos
LI	La primer etapa del cargador de arranque tiene habilitacion para cargar la segunda

	etapa del booteador, pero ha fallado al ejecutarlo. Esto puede estar causado por una no-coincidencia en la geometria del disco o porque se ha movido /boot/boot.b sin ejecutar el mapeador de LILO.
LIL	La segunda etapa arrancó pero no puede cargar el descriptor de la tabla de mapeo del disco. Esto es usualmente causado por una falla del disco o una no-coincidiencia en la geometria del mismo.
LIL?	La segunda etapa del booteador ha sido cargado con una direccion erronea, esto es usualmente causado por una no-coincidencia en la geometria del disco o porque se ha movido /boot/boot.b sin ejecutar el mapeador de LILO.
LIL-	El descriptor de la tabla de mapeo del disco esta corrupto, esto puede suceder por no-coincidencia en la geometria del disco o porque se ha movido /boot/map sin ejecutar el mapeador de LILO.
LILO	Todas las partes de LILO han sido correctamente cargadas

Si la BIOS anuncia un error mientras LiLo intenta cargar una imagen de arranque, la BIOS entregara un código. Este código esta entre los rangos 0x00 a 0xbb. Tendras que agenciarte del manual de usuario LILO para esta tarea.
“Extraído y traducido de The Linux Bootdisk HOWTO”

No Init Found

Si aparece el siguiente mensaje mientras estamos booteando la maquina cliente, mas precisamente, una vez que el disco RAM se ejecute:

```
Warning: unable to open an initial console
Kernel panic: no init found. Try passing init= option to kernel
```

Es porque el Kernel no encuentra el root path a @linuxrc, o sea al proceso padre 'INIT'.
Principalmente esto sucede cuando copiamos sin las debidas precauciones nuestro RootFileSystem de la maquina de desarrollo al Servidor.
Lo que sucedió es que ya sea cp o scp, rompio los enlaces blandos (o simbolicos), resultandole imposible al Kernel encontrar la ruta de Init.
Cerciorate de usar copia preservando los archivos y sus enlaces con las opciones '-a' y '-p'. Mas informacion en las man de 'cp' y 'scp'.

Cantidad de errores de compilacion

'make', 'make bzImage', 'make modules' o 'make modules_install' dan problemas de compilacion.
Debes ejecutar 'make mrproper' antes de hacer un make.

```
        bash# make mrproper
```

Si el problema persiste, entonces intenta con 'menuconfig' en vez de 'xconfig'. A veces la version GUI de xconfig causa algunos problemas:

```
        bash# export TERM=VT100
        bash# make menuconfig
```

Probemas clasicos de memoria una vez que arranca el Kernel

(extraido de la documentacion oficial del Kernel 2.4.22):

1) Hay algunas placas madres con bugs en BIOS que no pueden manejar apropiadamente mas alla de 16MB de RAM. Considera cambiar tu motherboard.

2) Hay ciertas motherboards que no cachean despues de cierta cantidad de memoria. Si tienes una de estas, vas a notar que tu sistema se pone mas lento, no mas rapido, cuando le agregas memoria. Considera cambiar tu motherboard.

Todos estos problemas pueden solucionarse, direccionando con la opcion a la hora de bootear "mem=XXXM"
(donde XXX es el tamaño de RAM a usar en megabytes).
Checa la doc de tu bootloader acerca de como pasarle parametros al kernel.

Hay otros problemas de memoria los cuales Linux no puede manejar.
Corrupcion de memoria al azar es sintoma de graves problemas a nivel hardware.
Intenta:

* Reducir la configuracion de memorioa en la BIOS a tiempos mas conservadores.

* Agregar un ventilador (cooling fan).

* No le hagas overclocking a tu CPU.

* Tener tus memorias checadas con un testeador de memorias. Considera usar memtest86.

* Cambiando tu CPU, cache, o motherboard por alguna que funcione.

* Desabilitando el cache de la BIOS.

* Intenta pasando el valor "mem=4M" en la opcion de kernel para limit ar a Linux solo usar muy poca cantidad de memoria.

Otros trucos:

* Intenta pasar la opcion "no-387" al kernel asi ignora algunos co-procesadores con bugs.

* Intenta pasando la opcion "no-hlt" para desabilitar la instrucción HLT potencialmente defectuosa (con un tremendo bug) en tu CPU.